SYNTHESIS OF BARRIER CERTIFICATE-BASED CONTROLLERS FOR SAFE ROBOTIC TASK EXECUTION

A Dissertation
Presented to
The Academic Faculty

By

Mohit Srinivasan

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
School of Electrical and Computer Engineering

Georgia Institute of Technology

December 2020

SYNTHESIS OF BARRIER CERTIFICATE-BASED CONTROLLERS FOR SAFE ROBOTIC TASK EXECUTION

Approved by:

Dr. Samuel Coogan, Advisor
Demetrius T. Paris Junior Professor
School of Electrical and Computer Engineering
School of Civil and Environmental Engineering
Georgia Institute of Technology

Dr. Magnus Egerstedt
Steve W. Chaddick School Chair and Professor
School of Electrical and Computer Engineering
Georgia Institute of Technology

Dr. Patricio Vela
Associate Professor
School of Electrical and Computer Engineering
Georgia Institute of Technology

Dr. Ye Zhao
Assistant Professor
School of Mechanical Engineering
Georgia Institute of Technology

Dr. Stefano Di Cairano
Distinguished Research Scientist, Senior Team Leader
Mitsubishi Electric Research Labs (MERL)

Date Approved: October 28, 2020

Till the full stop doesn't come, the sentence is not complete

M.S Dhoni

To Mythili, Vishwas, Aarohi, and my parents, Padma and Srinivasan

ACKNOWLEDGEMENTS

My deepest gratitude goes to my advisor, Dr. Sam Coogan, who decided to take me under his mentorship in Fall 2017, a year after my move to the US. Sam has always been encouraging of my ideas and plans throughout my PhD, and I will be forever indebted to him for the continuous support he has given me during the toughest of times. I have learned immensely from my conversations with him, and it has shaped the way I approach many questions in research. But most importantly, Sam has been the kindest human being and advisor which has been a source of strength for me. He nurtured an amazing lab environment and culture, and I am fortunate to have been a part of his journey. Thank you for everything, Sam. My sincerest gratitude also goes to Dr. Magnus Egerstedt, who I have had the pleasure of working with since 2017. I am extremely grateful that he provided me with the necessary foundation and opportunities to begin my PhD. He has always been a constant source of guidance and support for me, and I will always remain thankful for his presence in my life. I am also grateful to Dr. Patricio Vela for his constant support over the last few years, and for a lovely project collaboration. I have learned a lot from him over a short span of time, and I am glad he gave me an opportunity to collaborate with him.

I would also like to thank Dr. Stefano Di Cairano from Mitsubishi Electric Research Labs (MERL) for his support and guidance as my manager during my internship there. Dr. Ankush Chakrabarty and Dr. Rien Quirynen from MERL have also helped me tremendously by introducing me to new techniques to solve problems, and made my internship at MERL an extremely enjoyable experience. Dr. David Decker who was my manager at Landis+Gyr, has been a friend and mentor for me throughout the years. As an intern, he gave me the freedom to explore new ideas and avenues of research. Our lunch meetings were always fun and I am glad to have him as a source of support. My sincerest thanks to Jeff Scheb from Landis+Gyr who I met through the mentor jackets program at Georgia Tech. Jeff has been a constant source of support and help, especially during my initial

years in the US, and I am very grateful for his presence in my life. As I was navigating the unknown world of graduate school, it was Jeff who helped me explore various career opportunities and introduced me to the team at Landis+Gyr.

To my beloved colleagues in the FACTS Lab and GRITS Lab, and my delta-disk neighbors (Cesar, Max, Gustav, Matt, and Michael)- thank you for adding infinite amounts of fun to my PhD life! I cannot emphasize how important you all have been in my journey, and how grateful I am for your presence in my life. Max and Cesar have helped me get through some really tough times and have become a very important part of my life. They both made me feel at home during my stay in Atlanta. I am glad we met, and I hope we continue our amazing, fun, and stimulating conversations for all the years to come. Cheers to you guys! Thanks for making this ride much more enjoyable. A huge shout-out to my roommates Sagar and Varun for their priceless support over the years. Words cannot express how instrumental you have been in helping me finish my PhD. Nihaar has been like a brother to me ever since we met in kindergarten, and his support and conversations have helped me tremendously over the years. He has helped me get through some really tough times and I am very fortunate for his presence in my life. Thanks a lot bro! You are one of a kind.

I cannot emphasize how important my teachers from school, and undergrad have been in this journey. Dr. N.M. Singh from the Electrical Engineering department at VJTI, and Amaren Das from BARC have been extremely influential in my decision to pursue controls and robotics as a future research career. To all my teachers at OLPS High School, thank you for providing a solid foundation in my life. My math and physics teacher from school, Vivita Castelino inspired me to pursue a career in engineering and was always supportive of my passions. Santosh sir and Menon sir from SIES college, and Deb sir- thank you for being the pillars of strength during one of the most stressful periods of my life. I am where I am because of the immense efforts you all put in for my progress.

Without a doubt, this thesis would not have been possible without my parents. I have

put them through a lot of stress, but they have always been there for me no matter what the situation. Thank you for the twice-a-day phone calls that helped me stay motivated, and reduced the constant feeling of home-sickness. Mythili, Vishwas and Aarohi have also been instrumental in this journey and I cannot thank them enough for all the days when they cheered me up. This thesis is dedicated to all of you!

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

The objective of this thesis is to introduce and formalize a framework to guarantee task satisfaction for a robotic system, subject to a variety of constraints such as collision avoidance with obstacles, inter-agent collision avoidance in the case of swarm robotic type applications, connectivity constraints, and actuator constraints. In particular, this thesis is aimed at guaranteeing task satisfaction for complex specifications such as motion planning type problems for robotic systems such as autonomous vehicles and multi-robot systems. To do this, we make use of a recently developed tool known as control barrier functions. In addition, one of the major focus of this work is on the synthesis of controllers with real-time implementation capabilities that are amenable to deployment on actual robotic systems.

The synthesized controllers are embedded in an optimization framework known as a quadratic program, and hence feasibility of this program is an important issue that must be addressed when executing complex, safety-critical tasks. This thesis will provide techniques to address certain infeasibility scenarios which are not tackled in existing barrier function-based methods. In addition, existing literature asserts certain assumptions which are restrictive when dealing with complex task specifications. For example, a robot navigating an unknown environment has no knowledge of the obstacles in the domain. Hence, using the traditional barrier function-based framework is difficult since the unsafe regions are unknown. We will illustrate such scenarios, and provide frameworks to guarantee task satisfaction of the specification.

In addition to the algorithmic and theoretical contributions, we also implement our techniques on differential drive robots in order to validate the efficiency and advantages of the proposed methods. Algorithmic implementation insights and results are also discussed which will be useful when one considers such algorithms for deployment on large scale commercial autonomous platforms.

CHAPTER 1
INTRODUCTION

1.1 Motivation

Robotic systems such as the Boston dynamics SpotMini [1], Amazon's warehouse robots [2], or autonomous vehicles are tasked with executing complex behaviors while respecting a wide range of physical and environmental constraints. Hence, controllers with formal guarantees on task and constraint satisfaction are required for such applications. In particular, for safety-critical behaviors such as lane changing for autonomous vehicles, it is imperative that the safety constraints are satisfied during execution of the task, since otherwise it could lead to disastrous consequences. Similarly, motion planning tasks for systems such as robotic manipulators [3], personal assistants [4], and quadrotors [5] involves complex specifications to be satisfied by the system. Safety critical systems such as the power grid [6] and automation floors [2] rely on distributed controllers in order to function in the desired manner. These controllers are again tasked with satisfying complex specifications. Hence, failure of these controllers can lead to a collapse of the safety critical infrastructure. All these examples of existing technologies clearly illustrate the importance and need for control synthesis frameworks that guarantee safety, while satisfying the task at hand.

This dissertation will provide fundamental results aimed at task execution with constraint satisfaction. We provide techniques to synthesize controllers that guarantee task satisfaction without the need to perform computationally difficult methods such as abstractions of the state space. In particular, the emphasis is on the synthesis of controllers that are amenable to real-time implementation on actual robotic systems. To achieve this, we use tools from dynamical systems theory coupled with efficient convex optimization programs. This chapter will provide mathematical background on these tools.

1.2 Mathematical Background

As discussed earlier, the main topic of this thesis is on the notion of safe task execution for robotic systems. To that end, this section will provide background on the tools used to address this objective. In particular, we discuss methods to guarantee safety and reachability, the formal task specification language, and the structure of the optimization-based controller.

Control Barrier Functions

Consider a continuous time, control-affine dynamical system of the form

$$\dot{x} = f(x) + g(x)u \,, \tag{1.1}$$

where $f : \mathcal{D} \to \mathbb{R}^n$ and $g : \mathcal{D} \to \mathbb{R}^{n \times m}$ are locally Lipschitz continuous vector fields, $x \in \mathcal{D} \subset \mathbb{R}^n$ is the state of the system, and $u \in \mathbb{R}^m$ is the control input applied to the system. Here, $\mathcal{D}$ is the compact domain in which the system operates.

Now, consider a set $\mathcal{C} = \{x \in \mathcal{D} \mid h(x) \geq 0\}$ defined as the super-zero level set of a function h. Given an initial condition $x(0) \in \mathcal{C}$, we are interested in rendering $\mathcal{C}$ forward invariant. That is, the system trajectory is constrained to stay within $\mathcal{C}$ for all time. This is shown in Fig 1.1, where we see a robot that is avoiding an obstacle while trying to reach the desired goal region. In this case, the desired safe set is all points outside of the ellipsoidal obstacle. More formally, we are interested in guaranteeing safety for the system by ensuring that $x(t) \in \mathcal{C}$ for all $t \geq 0$.

Below, we formalize zeroing control barrier functions (ZCBFs), which guarantee forward invariance of a desired safe set in the state space. This tool will be used to address the safety constraints that are expected to be satisfied by the system. Before we introduce the notion of ZCBFs, we define an extended class $\mathcal{K}$ function [7] $\alpha : \mathbb{R} \to \mathbb{R}$ as a function that is strictly increasing and $\alpha(0) = 0$.

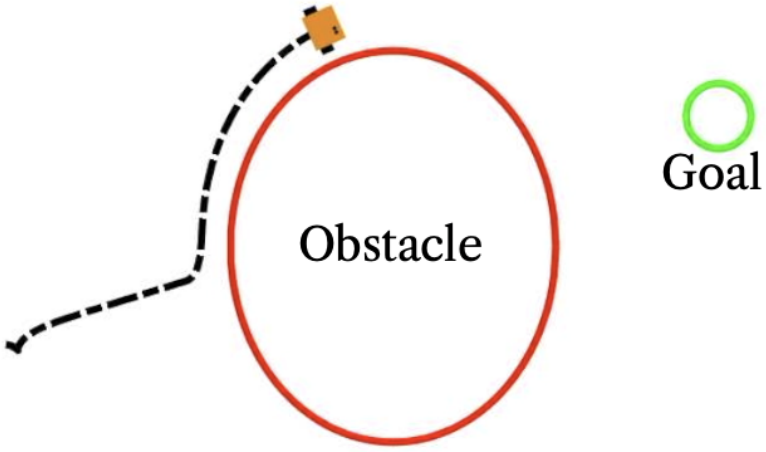


Figure 1.1: A differential drive robot moving towards a goal region, while avoiding the ellipsoidal obstacle. In this case, the set $\mathcal{C} = \{x \in \mathbb{R}^2 \mid h(x) \geq 0\}$ is the safe set i.e. the set of all positions of the robot outside the ellipse.

Definition 1 (Definition 3, [8])**.** *[Zeroing Control Barrier Function (ZCBF)] A continuously differentiable function $h : \mathcal{D} \to \mathbb{R}$ is a* zeroing control barrier function (ZCBF) *if there exists a locally Lipschitz extended class $\mathcal{K}$ function α such that for all $x \in \mathcal{D}$,*

$$\sup_{u \in \mathbb{R}^m} \Big\{ L_f h(x) + L_g h(x) u + \alpha(h(x)) \Big\} \geq 0 \tag{1.2}$$

where $L_f h(x) = \frac{\partial h(x)}{\partial x} f(x)$ and $L_g h(x) = \frac{\partial h(x)}{\partial x} g(x)$ are the Lie derivatives of h along f and g respectively. □

Let $\Sigma \subseteq \mathcal{D}$ be a safety set defined as $\Sigma = \{x \in \mathcal{D} \mid h(x) \geq 0\}$ where $h : \mathcal{D} \to \mathbb{R}$ is a ZCBF. The set of control inputs that satisfy (1.2) at any given state $x \in \mathcal{D}$ is then defined as

$$\mathcal{U}_\Sigma(x) = \Big\{ u \in \mathbb{R}^m \mid L_f h(x) + L_g h(x) u + \alpha(h(x)) \geq 0 \Big\}. \tag{1.3}$$

One can guarantee forward invariance of desired sets under the existence of a valid ZCBF as formalized in the following proposition.

Proposition 1 (Corollary 1, [8])**.** *If h is a ZCBF, then any continuous feedback controller satisfying $u \in \mathcal{U}_\Sigma(x)$ renders the set Σ forward invariant for the system* (1.1).

Proposition 1 provides necessary and sufficient conditions for constraining the system trajectory to stay within the safe set Σ. Typical forward invariance constraints that one can encounter when dealing with robotic systems include collision avoidance with obstacles, connectivity maintenance and inter-robot collision avoidance in swarms of robots, or automotive applications such as adaptive cruise control (ACC) where the vehicle must maintain a minimum safe distance and speed from the vehicle ahead.

In addition to guaranteeing safety, ZCBFs also guarantee asymptotic convergence to the safe set if the system starts outside the set. However, since we are interested in task execution and motion planning type applications where one could have a sequence of tasks that need to be executed in finite time, we require finite time convergence guarantees. Hence, we now define finite time convergence control barrier functions, first introduced in [9], which guarantee finite time convergence to desired sets in the state space.

Definition 2 (Finite Time Convergence Control Barrier Function (FCBF))**.** *A continuously differentiable function $h : \mathcal{D} \to \mathbb{R}$ is a* finite time convergence control barrier function *if there exist parameters $\rho \in [0, 1)$ and $\gamma > 0$ such that for all $x \in \mathcal{D}$,*

$$\sup_{u \in \mathbb{R}^m} \{L_f h(x) + L_g h(x) u + \gamma \cdot \textit{sign}(h(x)) \cdot |h(x)|^\rho\} \geq 0 \tag{1.4}$$

where $L_f h(x) = \frac{\partial h(x)}{\partial x} f(x)$ and $L_g h(x) = \frac{\partial h(x)}{\partial x} g(x)$. □

Let $\Gamma \subseteq \mathcal{D}$ be a target set defined as $\Gamma = \{x \in \mathcal{D} \mid h(x) \geq 0\}$ where $h : \mathcal{D} \to \mathbb{R}$. Let the set of control inputs that satisfy (1.4) at any state $x \in \mathcal{D}$ be given by

$$\mathcal{U}_\Gamma(x) = \left\{ u \in \mathbb{R}^m \middle| L_f h(x) + L_g h(x) u + \gamma \cdot \text{sign}(h(x)) \cdot |h(x)|^\rho \geq 0 \right\} \tag{1.5}$$

If h is a valid FCBF, then there exists a control input u that drives the state of the system

x to the target set $\{x \in \mathcal{D} \mid h(x) \geq 0\}$ in finite time, as formalized next.

Proposition 2 (Proposition III.1, [9]). *If h is a FCBF for* (1.1)*, then, for any initial condition $x_0 \in \mathcal{D}$ and any continuous feedback control $u : \mathcal{D} \to \mathbb{R}^m$ satisfying $u \in \mathcal{U}_\Gamma(x)$ for all $x \in \mathcal{D}$, the system will be driven to the set Γ in a finite time $0 < T < \infty$ such that $x(T) \in \Gamma$, where the time bound is given by $T = \frac{|h(x_0)|^{1-\rho}}{\gamma \cdot (1-\rho)}$. Moreover, Γ is forward invariant so that the system remains in Γ for all $t \geq T$.*

ZCBFs and FCBFs will form the basis for our control synthesis methodology. The advantage of the above results is that one can encode them in computationally efficient, convex optimization programs that will allow us to translate the above theoretical guarantees into actual robotic platforms. To that end, next we discuss the structure of the convex optimization program which uses the barrier functions as constraints.

Quadratic Program based controller

Given a FCBF or ZCBF h, the constraints (1.3) and (1.5) are affine in the control input u, and hence they can be conveniently encoded as affine constraints in a quadratic program (QP). Hence this formulation is amenable to efficient online computation of feasible control inputs, and thus a good control strategy that can be deployed on actual robotic platforms. In particular, for fixed $x \in \mathcal{D}$, the requirement that $u \in \mathcal{U}_\Gamma(x)$ and/or $u \in \mathcal{U}_\Sigma(x)$ becomes a linear constraint and we define a minimum power QP as

$$\begin{aligned} \min_{u \in \mathbb{R}^m} \quad & ||u||_2^2 \\ \text{s.t} \quad & u \in \mathcal{U}_\Gamma(x) \text{ and/or } u \in \mathcal{U}_\Sigma(x). \end{aligned} \tag{1.6}$$

We note that (1.6) can encode both finite time reachability as well as forward invariance requirements as constraints. This QP when solved, returns the point-wise in time, minimum power control law that drives the system to the goal set Γ in finite time and/or guarantees forward invariance of the safe set Σ. We will reference this idea of a QP based controller

throughout this thesis in the context of our theoretical framework.

Remark 1. *We note that multiple ZCBFs and multiple FCBFs can be encoded as separate constraints in the QP. In this case, we solve a single QP with multiple barrier function constraints. For example, see [8], [9].*

Given the controller structure, and the techniques to guarantee safety and reachability, the next step is to formally define the specification language which captures a wide range of robotic tasks that the system has to satisfy. This is detailed below.

Linear Temporal Logic

Complex and rich system properties can be expressed succinctly using linear temporal logic (LTL). The power of LTL lies in the wealth of tools available in the model checking literature [10] which can be leveraged for the synthesis of controllers in the continuous domain. LTL formulas are developed using atomic propositions which label regions of interest within the state space. These formulas are built using a specific grammar. LTL formulas without the next operator are given by the following grammar [10]:

$$\phi = \pi \mid \neg\phi \mid \phi \vee \phi \mid \phi \mathcal{U} \phi \tag{1.7}$$

where π is a member of the set of atomic propositions denoted by Π, and ϕ is a propositional formula that represents an LTL specification. Since we deal with continuous time systems in this work, the use of the "next" operator ($\bigcirc$) lacks meaningful interpretation, and hence, this operator is not included in our framework. Nonetheless, a large class of motion planning specifications (for example, the class of specifications proposed in [11]) do not require the next operator. In particular, finite time reachability specifications can be encoded in our proposed work.

We use the standard graphical notation for the temporal operators including $\Box$ ("Always"), $\Diamond$ ("Eventually"), $\Diamond\Box$ ("Persistence") and $\Box\Diamond$ ("Recurrence"). From the negation

($\neg$) and the disjunction ($\vee$) operators, we can define the conjunction ($\wedge$), implication ($\rightarrow$), and equivalence ($\leftrightarrow$) operators. We can thus derive for example, the eventually ($\Diamond$) and always ($\Box$) operators as $\Diamond\phi = \top\mathcal{U}\phi$ and $\Box\phi = \neg\Diamond\neg\phi$ respectively. Below we provide informal interpretations of these operators with respect to an LTL formula ϕ.

- $\Diamond\phi$ is satisfied if ϕ is satisfied sometime in the future. That is, ϕ is satisfied at some point of time in the future.
- $\Box\phi$ is satisfied if ϕ is satisfied for all time. That is, ϕ is satisfied for all time.
- $\Diamond\Box\phi$ is satisfied if ϕ becomes satisfied at some point of time in the future and then remains satisfied for all time.
- $\Box\Diamond\phi$ is satisfied if ϕ is satisfied infinitely often at various points of time in the future.

1.3 Thesis Outline

Below we provide an outline of the chapters in this thesis.

- **Chapter 3:** In this chapter, we introduce the framework for translating a given user defined specification into a sequence of barrier certificate-based controllers. The specification is formally defined in linear temporal logic (LTL). We describe a class of LTL specifications which capture a wide range of complex behaviours expected from a robotic system. Then, we provide algorithms which translate such a specification to a sequence of reachability objectives which are then encoded as barrier certificate-based controllers. We also provide formal guarantees that the synthesized trajectories from the framework satisfy the given task specification. We discuss techniques to smoothly transition between different reachability objectives, and provide guarantees on continuity of the control law as well
- **Chapter 4:** In this chapter, we illustrate a complex, multi-robot task where different behaviors of robots are encoded using barrier functions, and we propose a frame-

work that smoothly transitions between different graph structures required for each behavior. We discuss the key assumptions of this framework and the one presented in Chapter 3 which must be addressed when such techniques are used for safety-critical applications

- **Chapter 5:** Enforcing reachability and safety constraints using barrier certificates can prove to be strict in certain scenarios, and hence in this chapter, we provide two techniques to alleviate infeasibiliy scenarios associated with traditional control barrier function based methods. We provide motivating examples along with theoretical results to showcase the advantages of our proposed approaches

- **Chapter 6:** When dealing with complex systems with differential constraints (non-holonomic systems), then one has to account for such complexities in dynamics while synthesizing controllers. In this chapter, we introduce a new finite time control barrier function which can be used to satisfy both position and orientation reachability requirements for a differential drive robot. We show that traditional finite time barrier function techniques fail when considering such a system for this kind of task, and provide techniques to guarantee feasibility and task satisfaction for the system

- **Chapter 7:** Traditional control barrier functions guarantee safety for the system state, which is a point in the state space. However, in reality, robotic systems have a volume associated with them. Hence, in this chapter, we introduce "Extent-Compatible Control Barrier Functions" which guarantee safety for the shape of the robotic system in addition to its state. We provide two approaches to enforce such a safety constraints along with simulation and robotic implementation results

- **Chapter 8:** A key assumption that is prevalent in barrier functions literature is that the robotic system has complete knowledge of the safe sets i.e. complete knowledge of the barrier function that guarantees safety. However, this is a restrictive assumption as we show in this chapter with a simple path planning task. We thus propose a data-

driven approach wherein the robotic system learns the barrier function using sensory inputs from the environment. Implementation results are provided which show the efficiency of the proposed approach

CHAPTER 2
LITERATURE REVIEW

In this chapter, existing research and state-of-the art methods in control barrier functions, temporal logic based control of robotic systems, and machine learning based approaches to barrier functions, are discussed. We provide a brief history of control barrier functions, followed by a detailed literature survey.

2.1 Control Barrier Functions

Brief History

As motivated earlier, the requirement of safety is critical for all systems. Safety in the context of continuous time dynamical systems has been studied since the early 1940's. Nagumo in his seminal work [12] provided necessary and sufficient conditions for invariance of desired sets in the state space. In particular, given a dynamical system $\dot{x} = f(x)$ where $x \in \mathbb{R}^n$ is the state, the requirement that the system is forward invariant within a desired set $\mathcal{C} := \{x \in \mathcal{X} | h(x) \geq 0\}$ is satisfied by using Nagumo's theorem which provides necessary and sufficient conditions for set invariance by evaluating the time derivative of the smooth function h on the boundary of the set $\mathcal{C}$. That is,

$$\mathcal{C} \text{ is forward invariant } \iff \frac{dh(x)}{dt} \geq 0 \text{ for all } x \in \partial\mathcal{C} \tag{2.1}$$

where $\partial\mathcal{C} = \{x \in \mathbb{R}^n | h(x) = 0\}$ is the boundary of the desired set.

Then, in the early 2000's, "Barrier Certificates" were introduced in order to certify safety in the context of nonlinear [13] and hybrid systems [14]. The authors in [13] and [14] discussed the use of barrier functions in order to certify that a system would never enter an unsafe set in the state space. In particular, the authors also explored the necessity of barrier

certificates detailed in [15] and also discussed the application of barrier certificates in the stochastic framework [16].

This idea of barrier certificates was then extended onto the control theoretic setting by [17]. Given a safe set $\mathcal{S}$ in the state space defined by the smooth function h, the condition for safety is

$$\exists \quad u \in \mathbb{R}^m \text{ s.t } \frac{dh(x,u)}{dt} \geq 0 \implies \mathcal{S} \text{ is forward invariant} \tag{2.2}$$

Subsequently, the idea of uniting the stability property from Lyapunov functions and the invariance property from barrier functions was combined together to form the "Control Lyapunov Barrier Function" as discussed in [18]. However, the constraints in these works are much stronger than necessary. Hence, this led to the development of the modern version of control barrier functions introduced first in [19] and [20]. Given a safe set $\mathcal{C} \subset \mathbb{R}^n$, the barrier function constraint which enforces forward invariance is given by

$$\exists \quad u \in \mathbb{R}^m \text{ s.t } \frac{dh(x)}{dt} \geq -\alpha(h(x)) \implies \mathcal{C} \text{ is forward invariant} \tag{2.3}$$

where α is a locally Lipschitz extended class $\mathcal{K}$ function.

This modern version of the barrier function constraint is the one used in this thesis. Subsequent research in barrier functions is distributed over a wide range of domains ranging from automotive applications, multi-robot tasks, autonomous vehicles, and robotic manipulators. Below, we discuss literature pertinent to each of these applications.

Control Barrier Functions for Automotive Applications

Control barrier functions (CBFs) have been widely applied to problems in the area of automotive control. The authors in [19] relaxed the earlier formulation of the barrier certificate constraint (2.2). In particular, [19] first used the barrier function constraint in a quadratic program (QP) to generate the control action which guaranteed satisfaction of the constraint.

The QP formulation, by virtue of its convex cost and affine constraints in the control law, resulted in superior computational advantages. Subsequently, in [8], the authors introduced zeroing control barrier functions (ZCBFs) along with robustness properties associated with the safety set. That is, upon perturbation, the system converges back to the safe set asymptotically. Again, this was shown with a case study in the context of adaptive cruise control. The authors in [21] validate the previously described controllers on actual robotic hardware consisting of scaled down model cars. Driver assist features such as adaptive cruise control (ACC) are implemented in the car models. Recently, the authors in [22] have used CBFs to address the issue of autonomous vehicles merging at an intersection subject to safety constraints. The authors decompose the merging problem into a sequence of subproblems each characterized by a separate QP with barrier function constraints.

Control Barrier Functions for Robotic Systems

In the domain of robotics, control barrier functions (CBFs) have been used extensively in recent years. In particular, they have been used in the context of multi-agent systems to guarantee collision avoidance between robots [23], [24], [25]. Given a minimum distance to be maintained between the robots, the safety set is encoded as the super zero level-set of a zeroing control barrier function (ZCBF) [8]. The authors then use a QP based controller with the control barrier functions as affine constraints in the control law, in order to guarantee forward invariance of this safety set. This in turn implies that the robots never collide. Such a framework has also been applied to quadrotors [23] where the safety set is considered to be a super ellipsoid which allows quadrotors to avoid collisions. In [26], the authors combine ideas from reinforcement learning (RL) with discrete time barrier functions in order to control a brushbot. The authors in [27] also discuss the use of Sum of Squares (SOS) programming to synthesize control barrier functions which maximize the volume of the invariant set within which the system is rendered forward invariant. The authors in [28] address the issue of composability of multiple objectives using barrier func-

tions. In particular, the authors use tools from non-smooth analysis in order to guarantee the existence of a controller which satisfies the objectives for a multi-agent system. In our paper [29], barrier functions that guarantee safety not just for the system state (center of mass) of the robot but also its volume are introduced.

In addition to safety, finite time control barrier functions (FCBFs), first introduced in [9], guarantee finite time convergence to desired sets in the state space as opposed to zeroing control barrier functions which only guarantee asymptotic converge to desired sets. Recently, [9], [30], [31], [32], have explored finite time control barrier functions for finite time reachability specifications. In [9] and [30], finite time barrier functions were used to achieve smooth transitions between different behaviors in a multi-agent system. The key objective in [9] and [30] was to ensure composability of different behaviors in order to ensure that the graph formation is ready for the next desired behavior. In [32], the authors proposed a method for the composition of multiple finite time barrier functions in order to mitigate infeasibility issues associated with the QP based controller. The authors in [31] developed a fully automated framework for the control of mobile robots under temporal logic specifications, using barrier functions. Similar to [9], the authors in [33] use finite time barrier functions for achieving graph formations which are r-robust. This allows for the robots to achieve formations even in the presence of malicious robots.

Control barrier functions have also been used in the context of energy aware control for robotic swarms and prioritization of tasks in robotic swarms. In particular, [34] used control barrier functions in order to ensure that the energy levels of the system was never depleted beyond a certain minimum threshold. This allows for execution of robotic tasks over long periods of time during which conservation of energy levels is of key importance. In [35], the authors use barrier functions to develop a prioritization scheme for tasks to be performed by robotic swarms. This is useful in situations when one would require some robots to ignore tasks in order to conserve energy and later prioritize other tasks.

Very recently, the authors in [36] have used control barrier functions with reinforcement

learning in order to render systems safe during the learning process. However, the authors consider affine barrier functions and assume that the barrier is provided to the system. As part of our future work, we propose to overcome this limitation by using tools in machine learning to compute the barrier function for arbitrary sets in real time.

Control Barrier Functions for High Relative Degree Systems

Traditional barrier function formulations assume that the control input u shows up in the QP constraint (1.3) after taking one derivative of the barrier. That is, the authors in previously described papers assume that the barrier functions are of relative degree one. However, when using these techniques for complex system dynamics, especially with non-holonomic systems, this assumption proves to be restrictive. This is because when the control input u vanishes from the constraint (1.3), then there is no way to apply a control that guarantees safety. Hence, the authors in [37], introduce a technique based on back-stepping in order to generate barrier functions of high relative degree. Similarly, the authors in [38] introduce a framework for synthesizing higher order barrier functions which is more generalized than the previously discussed backstepping based paper. In this thesis, we discuss a specific example of an infeasibility scenario where the barrier function is not of relative degree one, and we utilize the structure of the proposed barrier function in order to alleviate this issue.

2.2 Temporal Logic based Control of Mobile Robots

Mobile robotic systems are expected to execute specifications which can be decomposed into finite horizon tasks (reachability, persistence) and infinite horizon tasks (safety, persistence, recurrence). In order to capture such specifications, linear temporal logic (LTL) is a commonly used specification language. Given a specification in LTL, one can synthesize robot trajectories using the tools available in model checking literature [10]. LTL based control of robotic systems has been of extensive research in recent years. Among existing works, one of the main steps in LTL based control of robots is the construction of a

finite abstraction of the original system (see [39], [40], [41], [42]). This abstraction is a graph that captures the different behaviors of the system. Given such an abstraction of the dynamical system and an LTL specification, controllers can be automatically constructed using an automata based approach [10], [43], [44], [41]. However, abstracting the system is computationally expensive especially with complex dynamics and specifications. Hence, methods to alleviate this expensive procedure have been explored. For example, [45], [46], [47] uses mixed integer linear programs (MILPs) in order to generate the control action by adopting an abstraction free technique. In the context of this thesis, we focus on abstraction free techniques for the control of robotic systems subject to temporal logic specifications.

Recently, barrier functions have been used in conjunction with temporal logic. The authors in [48] discuss the use of time varying control barrier functions for signal temporal logic tasks (STL). In [48], the authors use a discretization free approach. Time varying barrier functions are used for satisfying finite time reachability and invariance specifications. The authors in [49], [50], [51] propose control approaches for STL tasks. Discrete time systems are considered by the authors and the methods result, even for single-agent systems, in computationally difficult mixed integer linear programs (MILPs). Control frameworks for the discrete time non-deterministic setup have been presented in [52]. Learning based approaches coupled with temporal logic have appeared in [53], [54], [55]. The authors in [56] present a method to obtain satisfaction guarantees for continuous time multi-agent systems under a fragment of STL tasks. For single agent systems, such continuous time guarantees have been presented in [57], where a non-convex optimization problem may have to be solved.

2.3 Barrier Functions and Machine Learning

In recent years, there has been a growing interest in using tools from the machine learning community to address technology gaps in barrier function-based methods. Indeed, the advantages of model-free approaches help alleviate certain assumptions in existing literature.

In particular, the combination of learning based techniques with barrier functions has made their applicability more widespread and has garnered interest in the industry as well.

An imitation learning based approach was adopted by the authors in [58] in order to synthesize valid control barrier functions using trajectories and control inputs obtained from an expert provided dataset. The authors in [59] similarly use a neural network trained over a dataset of expert trajectories in order to synthesize valid barrier functions. These approaches however rely on a dataset of expert trajectories being available, which is quite restrictive, and does not necessarily hold in the real world. In contrast to the above approaches, the techniques discussed in this thesis are more suited for the case where an expert dataset is not available, and one has to synthesize the barrier functions during runtime. Barrier functions and machine learning was combined in a different context in the paper [60] where barrier functions guaranteed safety in a reinforcement learning framework during the exploration prcess. One of the major challenges in reinforcement learning is to guarantee safety for the system during the learning process, and hence barrier functions were rightfully used for this requirement. The authors in [61] used the Dataset Aggregation (DAgger) algorithm and proposed a novel approach for learning uncertainties in the system model which affects the evolution of a control barrier function. The functionality of the proposed approach was validated on a Segway system, both in simulation and in hardware.

Part I

Safe Robotic Task Execution

CHAPTER 3

FROM TASK SPECIFICATIONS TO CONTROLLERS: A TEMPORAL LOGIC AND BARRIER FUNCTIONS APPROACH

In this chapter, we present an architecture for the control of robotic systems subject to linear temporal logic specifications using control barrier functions, which addresses some of the challenges associated with the examples discussed in Chapter 1. In particular, we propose an abstraction-free framework which provides computational advantages over other temporal logic based control techniques. We provide an automatic framework to translate a given task into a sequence of barrier certificate-based controllers, and provide formal guarantees that the synthesized trajectories from the controller satisfy the given task.

Zeroing control barrier functions (ZCBFs) guarantee asymptotic convergence to desired sets [8]. However, since we focus on motion planning specifications, we require finite time reachability guarantees. Recently, [9], [32] have introduced finite time control barrier functions for finite time reachability specifications. In [9], finite time barrier functions were used to achieve smooth transitions between different behaviors in a multi-agent system. The key objective in [9] was to ensure composability of different formation behaviors by making sure that the multi-agent communication graph is appropriate for the next desired formation, whereas in [32], a method for the composition of multiple finite time barrier functions was introduced. Barrier functions have also been introduced in hybrid systems theory [62] to guarantee forward invariance of hybrid inclusions.

Finite and infinite horizon specifications which are useful for mobile robotic systems can be conveniently encoded using linear temporal logic (LTL). The power of LTL originates from the wealth of tools available in the model checking literature [10] which allows for generating trajectories for the robots given a specification in temporal logic. LTL based control of robotic systems has been well studied and standard methods first create a fi-

nite abstraction of the original dynamical system [39], [40], [41], [42]. This abstraction can informally be viewed as a labeled graph that represents possible behaviors of the system. Given such a finite abstraction, controllers can be automatically constructed using an automata-based approach [41], [10], [43], [44]. However, abstracting the state space is computationally expensive especially with complex system dynamics and specifications.

In the presented framework in this chapter, we avoid the difficulties associated with computation of any automaton from the specification or a discretization of the state space. Since CBFs can be conveniently encoded within a QP, the controller is amenable to real time implementations without the need for an abstraction of the state space or the system dynamics. Other authors have explored discretization free techniques as well. The authors in [48] discuss the use of time varying control barrier functions for signal temporal logic tasks (STL). In [63], the authors use time-varying barrier functions for control of coupled multi-agent systems subject to STL tasks. In both [48] and [63], the authors do not allow for repetitive tasks, a specification which can be captured by our proposed framework. The authors in [49], [50], [51] discuss control methods for STL tasks. However, the methods proposed result in computationally expensive mixed integer linear programs. Control methods in the discrete time non-deterministic setting have been explored by [52]. Learning based frameworks are discussed by the authors in [53], [54], [55]. Control techniques for continuous-time multi-agent systems given fragment of STL tasks has been presented in [56]. The authors in [57] discuss a similar continuous time method. However, a non-convex optimization problem may have to be solved.

In this chapter, we propose a barrier function-based controller framework to synthesize system trajectories that satisfy a given user defined specification. In particular, the proposed framework automatically translates the user defined specification formalized in a subset of LTL to a sequence of barrier function-based quadratic programs. The approach adopted in this chapter is a discretization free approach which alleviates some of the computational issues arising from abstraction based control of mobile robots [39, 40, 41, 42, 43, 44]. Then,

we provide formal guarantees that the proposed controller framework produces a system trajectory that satisfies the given specification. The proposed family of LTL specifications in our work can capture more complex specifications than the fragment considered in [48, 63]. In addition, our guarantees are different from other chapters on temporal logic based control using barrier functions [48, 63] in that we characterize the family of trajectories that satisfy the given specification, and then prove that the proposed controller indeed produces a trajectory that belongs to the set of satisfying trajectories. The trajectory generated by the proposed CBF based controller is analyzed and the guarantees of CBFs translate to guarantees on the system trajectory.

3.1 Problem Setup

We consider a continuous time mobile robotic system in control-affine form as in (1.1). We assume that the domain $\mathcal{X}$ contains regions of interest which are labeled by a set of atomic propositions $\Pi = \{\pi_1, \pi_2, \pi_3, \ldots, \pi_n\}$ with the labeling function $L : \mathcal{X} \to 2^{\Pi}$ so that $\pi \in \Pi$ is true at $x \in \mathcal{X}$ if and only if $\pi \in L(x)$. These regions may overlap and need not constitute a partition or cover of $\mathcal{X}$. For each $\sigma \in 2^{\Pi}$, we have $L^{-1}(\sigma) = \{x \in \mathcal{X} \mid \sigma = L(x)\}$. Let $\Pi_{aug} = \{\pi_1, \pi_2, \ldots, \pi_n, \overline{\pi_1}, \overline{\pi_2}, \ldots, \overline{\pi_n}\}$ be the augmented set of atomic propositions where we define $\overline{\pi_i} = \neg\pi_i$ for all $i \in \{1, 2, \ldots, n\}$. The set Π_{aug} is also called the set of *literals* [10]. Thus, we identify $\neg\overline{\pi}_i = \pi_i$ for all $i \in \{1, 2, \ldots, n\}$. In addition, define

$$S(\Pi_{aug}) = \{J \subset \Pi_{aug} \,|\pi \in J \implies \neg\pi \notin J \text{ for all } \pi \in \Pi_{aug}\} \tag{3.1}$$

$$P(\Pi_{aug}) = \{J \subset \Pi_{aug} \mid (\pi_i \in J) \oplus (\overline{\pi_i} \in J) \text{ for all } i \in \{1, 2, \ldots, n\}\} \tag{3.2}$$

where $\oplus$ is the exclusive disjunction operator. Observe that $P(\Pi_{aug}) \subset S(\Pi_{aug})$. A subset of Π_{aug} belongs to the family $S(\Pi_{aug})$ if it does not contain an atomic proposition and its negation simultaneously, and it further belongs to $P(\Pi_{aug})$ if it contains each atomic proposition exclusive-or its negation.

We consider a fragment of LTL, denoted by $LTL_{robotic}$, which is a modification of the fragment considered in [64]. Our proposed fragment covers a large class of motion planning tasks, such as the ones discussed in [11], expected from a robotic system.

Definition 3 (Fragment of LTL)**.** *The fragment $LTL_{robotic}$ is defined as the class of LTL specifications of the form*

$$\phi = \phi_{globe} \wedge \phi_{reach} \wedge \phi_{rec} \wedge \phi_{act} \tag{3.3}$$

where $\phi_{globe} = \Box\psi_1$, $\phi_{reach} = \bigwedge_{j\in\mathcal{I}_2} \Diamond\psi_2^j$, $\phi_{rec} = \bigwedge_{j\in\mathcal{I}_3} \Box\Diamond\psi_3^j$ and $\phi_{act} = \Diamond\Box\psi_4$. Here $\mathcal{I}_2$ and $\mathcal{I}_3$ are finite index sets and ψ_1, ψ_2^j for all j, ψ_3^j for all j and ψ_4 are propositional formulas of the form $\psi_i = \bigwedge_{\pi\in J_i} \pi$ with $J_i \in S(\Pi_{aug})$ for all $i \in \{1, 4\}$, $\psi_i^j = \bigwedge_{\pi\in J_i^j} \pi$ with $J_i^j \in S(\Pi_{aug})$ for all $i \in \{2, 3\}$ and for all $j \in \mathcal{I}_i$. □

Below we provide informal definitions of the specifications appearing in the above definition.

- ϕ_{globe}: This type of specification captures properties that must hold throughout the execution of the system. For example, collision avoidance with obstacles must hold at all times when a robot is navigating in the workspace.

- ϕ_{reach}: Specifications of this form capture finite time reachability requirements for the system. For example, a robot must reach a region of interest within a finite time.

- ϕ_{rec}: This recurrence specification captures, for instance, scenarios where the system must visit regions infinitely often. For example, a robot must visit room A and room B infinitely often.

- ϕ_{act}: This type of specification captures persistence requirements. For example, a robot must reach a region and then stay within the region for all time.

As compared to [64], we additionally incorporate reachability specifications ($\diamond$) without increasing the system complexity due to the abstraction free nature of our proposed framework. We do not include response-to-environment specifications ($\square(A \implies \diamond B)$) since time-varying or reactive system specifications are not considered in the context of our proposed barrier function framework. With regard to other widely used fragments, our proposed fragment allows for persistence ($\diamond\square$) which cannot be expressed by the Generalized Reactivity (GR(1)) fragment or computation tree logic (CTL) [65]. Our proposed fragment also allows for repetitive tasks (ϕ_{act} and ϕ_{rec}) which cannot be captured by the fragment considered in very recent work on barrier function based control using temporal logic [48, 63]. A wide range of robotic tasks such as path planning, environmental monitoring, long-duration autonomy, and coverage can be expressed using our proposed fragment.

For any propositional formula ψ omitting temporal operators (*e.g.*, a conjunction of literals), we define the *proposition set*, denoted $[\![\psi]\!]$, as the set of all states that satisfy ψ. That is,

$$[\![\psi]\!] = \{x \in \mathcal{X} \mid L(x) \vDash \psi\} \tag{3.4}$$

where $L(x) \vDash \psi$ signifies that ψ is true under the evaluation for which all and only propositions in $L(x) \subset \Pi$ are true.

We assume that for each atomic proposition $\pi \in \Pi$, there exists a continuously differentiable function $h : \mathcal{X} \to \mathbb{R}$ such that $[\![\pi]\!] = \{x \in \mathcal{X} | h_\pi(x) \geq 0\}$. In this chapter, similar to the assumption in [8], we assume that $L_g h_\pi(x) \neq 0$ for all $x \in \mathcal{X}$. We ignore the measure-zero set $\{x \in \mathcal{X} | h_\pi(x) = 0\}$, and identify $[\![\overline{\pi}]\!] = \{x \in \mathcal{X} | h_\pi(x) < 0\}$ for each $\pi \in \Pi$. Thus we define $h_{\overline{\pi}}(x) = -h_\pi(x)$ for all $\pi \in \Pi$.

The fragment $LTL_{robotic}$ encompasses a class of specifications which cover properties

such as finite time reachability, persistence, recurrence, and invariance. These properties are useful to express a number of common robotic system specifications.

Recall that for any $\sigma \in 2^{\Pi}$, $L^{-1}(\sigma) = \{x \in \mathcal{X} | \sigma = L(x)\}$. We define a *trace* as a sequence of sets of atomic propositions. The *trace of a trajectory* $x(t)$ of a continuous time dynamical system is defined as the sequence of propositions satisfied by the trajectory. As formalized in [66], an infinite sequence $\sigma = \sigma_0\sigma_1 \ldots$ where $\sigma_i \subseteq \Pi$ for all $i \in \mathbb{N}$ is the trace of a trajectory $x(t)$ if there exists an associated sequence $t_0t_1t_2 \ldots$ of time instances such that $t_0 = 0$, $t_k \to \infty$ as $k \to \infty$ and for each $m \in \mathbb{N}$, $t_m \in \mathbb{R}_{\geq 0}$ satisfies the following conditions:

- $t_m < t_{m+1}$
- $x(t_m) \in L^{-1}(\sigma_m)$
- If $\sigma_m \neq \sigma_{m+1}$, then for some $t_m^{'} \in [t_m, t_{m+1}]$, $x(t) \in L^{-1}(\sigma_m)$ for all $t \in (t_m, t_m')$, $x(t) \in L^{-1}(\sigma_{m+1})$ for all $t \in (t_m', t_{m+1})$, and either $x(t_m') \in L^{-1}(\sigma_m)$ or $x(t_m') \in L^{-1}(\sigma_{m+1})$.
- If $\sigma_m = \sigma_{m+1}$ for some m, then $\sigma_m = \sigma_{m+k}$ for all $k > 0$ and $x(t) \in L^{-1}(\sigma_m)$ for all $t \geq t_m$.

The last condition of the above definition implies that a trace contains a repeated set of atomic propositions only if this set is then repeated infinitely often. This is useful to capture for example, a stability condition of the system. By forbidding repetitions in other cases, we ensure that a particular trajectory possesses a unique trace. This exclusion is without loss of generality since we only consider $LTL_{robotic}$ specifications without the next operator. We now define the problem statement that is addressed in this chapter.

Problem Statement 1. *Given a specification in $LTL_{robotic}$ as in* (3.3) *which is to be satisfied by a continuous time robotic system with dynamics as in* (1.1)*, synthesize a point-wise minimum norm controller as in* (1.6) *which produces a system trajectory whose trace satisfies the given specification.*

We are interested in generating system trajectories using controllers of the form (1.6), which guarantee satisfaction of the given $\text{LTL}_{\text{robotic}}$ specification. As a secondary objective, we are interested in achieving the stated task by expending minimum power, point-wise in time. These objectives are captured by the problem statement. Note that a key requirement in order to address the above objective is feasibility of the controller. This is taken as an assumption in this chapter, however we touch upon this and other standard assumptions in subsequent chapters of the thesis.

3.2 Synthesis and Analysis of Quadratic Program-Based Controller

In this section, we detail the theoretical framework which provides formal guarantees that the quadratic program (QP) based controller indeed produces a system trajectory that satisfies the given specification. We also describe the methodology to synthesize the barrier funtion based QP controller given an $LTL_{robotic}$ specification.

3.2.1 Lasso Type Constrained Reachability Objectives

It is well established that if there exists a trace that satisfies a specification belonging to the fragment LTL_{robotic}, then there exists a trace which satisfies the specification in *lasso* or *prefix-suffix* form ([10], pp 272), where a trace σ is in lasso form if it is comprised of a finite horizon prefix σ_{pre} and a finite horizon suffix σ_{suff} that is repeated infinitely often. Both σ_{pre} and σ_{suff} are finite sequences of sets of atomic propositions such that the trace σ is equal to the prefix followed by an infinite repetition of the suffix. Such a lasso-type trace is denoted as $\sigma = \sigma_{pre}(\sigma_{\text{suff}})^{\omega}$, where ω denotes infinite repetition. Atomic propositions of a continuous time dynamical system are subsets of the domain, and, hence, it is possible to interpret such lasso traces as sequences of constrained reachability problems in lasso form, which forms the basis of our control synthesis methodology. This is formalized in the following definitions.

Definition 4 (Constrained reachability objective)**.** *Given a target set $\Gamma \subset \mathcal{X}$ and a safety set $\Sigma \subset \mathcal{X}$, the* constrained reachability objective*, denoted by $R(\Sigma, \Gamma)$, is defined as the reachability problem to be solved so that the state of the system reaches the set Γ in finite time while remaining in Σ until it reaches Γ.* □

The constrained reachability objective for a system (1.1) is solved from a given initial condition in Σ if a control policy is found which drives the state of the system to Γ while remaining in Σ until it reaches Γ. For example, a reachability objective denoted by $R(B, A)$ signifies that the system must reach region A in finite time while staying in region B until it reaches region A. The constrained reachability objective implies finding a control policy that solves the above objective successfully. The main intuition of our framework is that given a reachability objective $R(\Sigma, \Gamma)$, one can encode the safety set Σ using zeroing barrier functions, and the target set Γ using finite time barrier functions.

In order to encode complex objectives such as recurrence and persistence, we need to capture repetitive behavior, and hence, below we formally define a "Lasso Type Constrained Reachability Sequence" which can enncode complex objectives as a sequence of reach-avoid problems.

Definition 5 (Lasso Type Constrained Reachability Sequence)**.** *A* lasso-type constrained reachability sequence *is a sequence of constrained reachability objectives in lasso form such that each subsequent safety set is compatible with the prior goal set. That is, a lasso-type constrained reachability sequence has the form*

$$\mathcal{R}_{lasso} = \Big(R_1 R_2 \dots R_p\Big)\Big(R_{p+1}, R_{p+2} \dots R_{p+\ell}\Big)^{\omega}, \tag{3.5}$$

where $p > 0$, $\ell \geq 1$, and each $R_j = R(\Sigma_j, \Gamma_j)$ for some $\Gamma_j, \Sigma_j \subset \mathcal{X}$ satisfying $\Gamma_j \subseteq \Sigma_{j+1}$ for all $j \in \{1, 2, \dots, p + \ell - 1\}$ and $\Gamma_{p+\ell} \subseteq \Sigma_{p+1}$. The sequence $(R_1 R_2 \dots R_p)$ is a finite horizon prefix objective and $(R_{p+1}, R_{p+2} \dots R_{p+\ell})$ is a finite horizon suffix objective that is repeated infinitely often. □

The lasso-type constrained reachability sequence is considered feasible if each constituent reachability objective is solved successfully in sequence. For example, consider the task specification, "The robot must first visit region A, then region B, while avoiding the obstacle C. The lasso-type reachability sequence that satisfies this task is given by

$$\mathcal{R}_{\text{lasso}} = \Big(R_1 R_2\Big)\Big(R_3\Big)^{\omega}$$

where $R_1 = R(\mathrm{C}, \mathrm{A})$, $R_2 = R(\mathrm{C}, \mathrm{B})$ and $R_3 = R(\mathrm{C}, \varnothing)$. Note that if $p = 0$, then the finite prefix has length zero and the lasso sequence is then given by

$$\mathcal{R}_{lasso} = \Big(R_1, R_2 \dots R_\ell\Big)^{\omega}. \tag{3.6}$$

A reachability objective R_i is said to be solved successfully if the state of the system $x(t) \in \Gamma_i \cap \Sigma_i$ for some $t \in (0, \infty)$ and for any $i \in \{1, 2, \dots, p + \ell\}$.

By the preceding discussion, if there exists a trace that satisfies a given $LTL_{robotic}$ specification, then there exists a lasso-type constrained reachability sequence which, if feasible, guarantees that the system satisfies the $LTL_{robotic}$ specification. One can view the lasso type reachability sequence as a bridge between the $LTL_{robotic}$ specification and the set based approach of our proposed controller. The lasso-type reachability sequence is akin to lasso runs that one would encounter in an automata-based approach to control synthesis.

3.2.2 Construction Of Lasso-type Reachability Sequence

Consider a $LTL_{robotic}$ specification ϕ as in (3.3). Given ϕ, our first objective is to generate the lasso-type constrained reachability sequence of the form (3.5). To do this, we formally define a *lasso template*. Given a $LTL_{robotic}$ specification ϕ, a *lasso template* is an

enumeration of the form

$$\mathcal{O}_2 : \{1, 2, \ldots, k\} \to \mathcal{I}_2 \tag{3.7}$$

$$\mathcal{O}_3 : \{1, 2, \ldots, \ell\} \to \mathcal{I}_3 \tag{3.8}$$

where the index sets $\mathcal{I}_2$ and $\mathcal{I}_3$ are as per Definition 3 and $k = |\mathcal{I}_2|$ and $\ell = \max\{|\mathcal{I}_3|, 1\}$.

Note that it is computationally straightforward to obtain *some* lasso template simply by arbitrarily enumerating the elements of the index sets $\mathcal{I}_2$ and $\mathcal{I}_3$. A lasso-type reachability sequence of the form (3.5) or (3.6) is constructed using Algorithm 1. The LTL_{robotic} specification, and the lasso template $\mathcal{O}_2$ and $\mathcal{O}_3$ are the inputs to Algorithm 1. The output is a lasso-type constrained reachability sequence of the form (3.5) or (3.6).

3.2.3 Synthesis of Quadratic Program based Controller

We next encode the reachability objectives as finite time and zeroing control barrier functions in a QP. This is described in Algorithm 2. Each Γ_i is encoded with FCBFs with (1.5) or (5.6) as constraint(s) whereas each Σ_i is encoded with ZCBFs with (1.3) as constraint(s) in the QP. The designer is free to choose a locally Lipschitz α function for (1.3). In order to solve a particular reachability objective $R_i(\Sigma_i, \Gamma_i)$ where $i \in \{1, 2, \ldots, n\}$, we solve a QP as in (1.6). Note that solving a QP in real time is typically done in a few milliseconds, and hence Algorithm 2 is amenable to real time implementation on robotic platforms. The next step is to provide theoretical guarantees that the trajectory generated by the barrier certificates based controller in Algorithm 2 indeed satisfies the specification. The next section builds the required machinery and provides a formal guarantee that if the QP from Algorithm 2 is feasible, then the trace of the system trajectory satisfies the specification.

Algorithm 1 Lasso-type Reachability Sequence Generator

Input : ϕ**,** $\mathcal{O}_2$**,** $\mathcal{O}_3$
Output: $\mathcal{R}_{lasso}$

if $J_4 \neq \varnothing$ **then**
  $p \leftarrow k+1$
  if $k \neq 0$ **then**
    $\Gamma_i = [\![\psi_2^{\mathcal{O}_2(i)}]\!]$ for all $i = 1, 2, \ldots, p-1$
    $\Sigma_i = [\![\psi_1]\!]$ for all $i = 1, 2, \ldots, p-1$
  end if
  $\Gamma_p = [\![\psi_4]\!]$
  $\Sigma_p = [\![\psi_1]\!]$
  $\Gamma_{p+i} = [\![\psi_3^{\mathcal{O}_3(i)}]\!]$ for all $i = 1, 2, \ldots, \ell$
  if $J_1 = \varnothing$ **then**
    $\Sigma_{p+i} = [\![\psi_4]\!]$ for all $i = 1, 2, \ldots, \ell$
  else
    $\Sigma_{p+i} = [\![\psi_1]\!] \cap [\![\psi_4]\!]$ for all $i = 1, 2, \ldots, \ell$
  end if
  $R_i = R_i(\Sigma_i, \Gamma_i)$ for all $i = 1, 2, \ldots, p+\ell$
  return $\mathcal{R}_{lasso}$ as in (3.5)
else
  $p \leftarrow k$
  if $p \neq 0$ **then**
    $\Gamma_i = [\![\psi_2^{\mathcal{O}_2(i)}]\!]$ for all $i = 1, 2, \ldots, p$
    $\Gamma_{p+i} = [\![\psi_3^{\mathcal{O}_3(i)}]\!]$ for all $i = 1, 2, \ldots, \ell$
    $\Sigma_i = [\![\psi_1]\!]$ for all $i = 1, 2, \ldots, p+\ell$
    $R_i = R_i(\Sigma_i, \Gamma_i)$ for all $i = 1, 2, \ldots, p+\ell$
    return $\mathcal{R}_{lasso}$ as in (3.5)
  else
    $\Gamma_{p+i} = [\![\psi_3^{\mathcal{O}_3(i)}]\!]$ for all $i = 1, 2, \ldots, \ell$
    $\Sigma_{p+i} = [\![\psi_1]\!]$ for all $i = 1, 2, \ldots, \ell$
    $R_i = R_i(\Sigma_i, \Gamma_i)$ for all $i = 1, 2, \ldots, p+\ell$
    return $\mathcal{R}_{lasso}$ as in (3.6)
  end if
end if

Algorithm 2 Quadratic Program based Controller

Input : $\mathcal{R}_{lasso}$

if $p \neq 0$ **then**
  for $i = 1, 2, \ldots, p$ **do**
    Encode Γ_i with FCBFs
    Encode Σ_i with ZCBFs
    while $x \notin \Gamma_i$ **do**
      Solve $R(\Sigma_i, \Gamma_i)$ as in (1.6)
    end while
  end for
end if
while true **do**
  for $i = p + 1, \ldots, p + \ell$ **do**
    Encode Γ_i with FCBFs
    Encode Σ_i with ZCBFs
    while $x \notin \Gamma_i$ **do**
      Solve $R(\Sigma_i, \Gamma_i)$ as in (1.6)
    end while
  end for
end while

3.2.4 Analysis Of Trajectory Generated by QP Controller

Observe that there is a one-to-one correspondence between elements of $P(\Pi_{aug})$ and subsets of Π. Let $\iota : 2^{\Pi} \to P(\Pi_{aug}) \subset 2^{\Pi_{aug}}$ be the canonical bijective mapping for a subset $\sigma \in 2^{\Pi}$ with the corresponding mapping $\iota(\sigma) \in P(\Pi_{aug})$ given by,

$$\pi \in \sigma \iff \pi \in \iota(\sigma) \text{ and } \pi \notin \sigma \iff \overline{\pi} \in \iota(\sigma). \tag{3.9}$$

For notational convenience, we do not explicitly differentiate between a subset $\sigma \subset \Pi$ and its mapping $\iota(\sigma) \in P(\Pi_{aug})$.

Given Algorithm 2, we now provide formal guarantees which prove that the QP from Algorithm 2 indeed produces a system trajectory which satisfies the system specification. This is done by leveraging the formalism of LTL and the guarantees of control barrier functions. To that end, below we formalize the set of all system trajectories that satisfy a given LTL specification, based on the user define lasso template (3.7).

Definition 6 (Descendant)**.** *Given a $LTL_{robotic}$ specification ϕ with a lasso template $\mathcal{O}_2$ and $\mathcal{O}_3$, a* descendant *of the lasso template is any infinite length sequence of the form*

$$\sigma = \left\{\sigma_{1,1}\sigma_{1,2}\dots\sigma_{1,n_1}\right\}\left\{\sigma_{2,1}\sigma_{2,2}\dots\sigma_{2,n_2}\right\}\dots\left\{\sigma_{p,1}\sigma_{p,2}\dots\sigma_{p,n_p}\right\}\dots, \quad (3.10)$$

where $\sigma_{i,j} \in P(\Pi_{aug})$ for all $i = 1, 2, \dots$, $j = 1, 2, \dots, n_i$ and

1. *$J_1 \subseteq \sigma_{i,j}$ for all $i \in \{1, 2, \dots, p\}$ and for all $j \in \{1, 2, \dots, n_i\}$*
2. *$J_2^{\mathcal{O}_2(i)} \subseteq \sigma_{\mathcal{O}_2(i),n_{\mathcal{O}_2(i)}}$ for all $i \in \{1, 2, \dots, k\}$*
3. *$J_4 \subseteq \sigma_{p,n_p}$*
4. *$J_3^{\mathcal{O}_3(i)} \subseteq \sigma_{m,n_m}$ where $m = p + d\ell + \mathcal{O}_3(i)$ for all $d \in \{0, 1, 2\dots\}$ and for all $i \in \{1, 2, \dots, \ell\}$*
5. *$J_1 \cup J_4 \subseteq \sigma_{i,j}$ for all $i \in \{p+1, \dots\}$ and for all $j \in \{1, 2, \dots, n_i\}$.* □

Intuitively, a descendant σ of a given template is a sequence of atomic propositions visited by the system such that it respects the safety sets Σ_i and also reaches the target sets Γ_i in a finite time for all $i \in \{1, 2, \dots, p+l\}$. Consider the example discussed before, where a robot must first visit region A, then region B while avoiding the obstacle C. The lasso template for this task is $\mathcal{O}_2(1) = A$, $\mathcal{O}_2(2) = B$. Given this template, one valid instantiation of the descendant (3.10) is

$$\sigma = \left\{\{\bar{A}, \bar{B}, \bar{C}\}\{A, \bar{B}, \bar{C}\}\right\}\left\{\{\bar{A}, \bar{B}, \bar{C}\}\{\bar{A}, B, \bar{C}\}\right\}\left\{\{\bar{A}, B, \bar{C}\}\right\}^{\omega},$$

which satisfies the conditions of Definition 6. In (3.10), each set $\sigma_i = \{\sigma_{i,1}\sigma_{i,2}\dots\sigma_{i,n_i}\}$ corresponds to the i^{th} constrained reachability objective in the lasso sequence (3.5) or (3.6) and the set $\sigma_p = \{\sigma_{p,1}\sigma_{p,2}\dots\sigma_{p,n_p}\}$ is the last constrained reachability objective in the finite prefix part of the lasso sequence after which the sequence switches to the suffix.

Proposition 3. *Given a lasso template for a $LTL_{robotic}$ specification ϕ as in* (3.3)*, any descendant σ of this template is such that $\sigma \vDash \phi$.*

Proof. Let $\phi = \phi_{globe} \wedge \phi_{reach} \wedge \phi_{rec} \wedge \phi_{act}$ be a specification as in (3.3). Let $\mathcal{O}_2$ and $\mathcal{O}_3$ be a lasso template for the specification. Let σ be a descendant of the lasso template as in Definition 6.

We provide a proof by construction by considering four individual cases for the specification ϕ. Then, since conjunction preserves the results from these cases (Fig 5.2, pp 236 [10]), we combine them to provide a proof for the entire fragment of LTL.

Case 1: Suppose $\phi = \phi_{globe} = \square\psi_1$ for $\psi_1 = \bigwedge_{m=1}^{n} \pi_m$, where $\pi_m \in \Pi_{aug}$. Thus we have $J_1 = \{\pi_1, \ldots, \pi_n\}$, $J_2^{\mathcal{O}_2(i)} = \{\varnothing\}$ for all $i \in \{1, 2, \ldots, k\}$, $J_3^{\mathcal{O}_3(i)} = \{\varnothing\}$ for all $i \in \{1, 2, \ldots, \ell\}$ and $J_4 = \{\varnothing\}$. A descendant trace of the template is as per Definition 6. Thus, from condition 1 in Definition 6 , we observe that $J_1 = \{\pi_1, \ldots, \pi_n\} \subseteq \sigma_{i,j}$ for all $i \in \{1, 2, \ldots\}$ and for all $j \in \{1, 2, \ldots, n_i\}$. Hence, we can conclude that $\sigma \vDash \phi_{globe}$.

Case 2: Suppose $\phi = \phi_{act} = \diamond\square\psi_4$ for $\psi_4 = \bigwedge_{m=1}^{n} \pi_m$ where $\pi_m \in \Pi_{aug}$. Thus we have $J_1 = \{\varnothing\}$, $J_2^{\mathcal{O}_2(i)} = \{\varnothing\}$ for all $i \in \{1, 2, \ldots, k\}$, $J_3^{\mathcal{O}_3(i)} = \{\varnothing\}$ for all $i \in \{1, 2, \ldots, \ell\}$ and $J_4 = \{\pi_1, \ldots, \pi_n\}$. A descendant trace of the template has a closed form expression as in Definition 6. Thus, from condition 3 in Definition 6, we have $J_4 \subseteq \sigma_{p,n_p}$, and from condition 5 in Definition 6, we observe that $J_4 = \{\pi_1, \ldots, \pi_n\} \subseteq \sigma_{i,j}$ for all $i \in \{p+1, p+2, \ldots\}$ and for all $j \in \{1, 2, \ldots, n_i\}$. Hence, we can conclude that $\sigma \vDash \phi_{act}$.

Case 3: Suppose $\phi = \phi_{reach} = \bigwedge_{j \in \mathcal{I}_2} \diamond\psi_2^j$. Thus we have $J_1 = \{\varnothing\}$, $J_3^{\mathcal{O}_3(i)} = \{\varnothing\}$ for all $i \in \{1, 2, \ldots, \ell\}$ and $J_4 = \{\varnothing\}$. A descendant trace of the template has a closed form expression as in Definition 6. Thus, from condition 2 in the definition, we observe that $J_2^{\mathcal{O}_2(m)} = \{\pi_1, \ldots, \pi_n\} \subseteq \sigma_{\mathcal{O}_2(m), n_{\mathcal{O}_2(m)}}$ for all $m \in \{1, 2, \ldots, k\}$. Hence, we can conclude that $\sigma \vDash \phi_{reach}$.

Case 4: Suppose $\phi = \phi_{rec} = \bigwedge_{j \in \mathcal{I}_3} \square\diamond\psi_3^j$. Thus we have $J_1 = \{\varnothing\}$, $J_2^{\mathcal{O}_2(i)} = \{\varnothing\}$ for all $i \in \{1, 2, \ldots, k\}$ and $J_4 = \{\varnothing\}$. A descendant trace of the template has a closed form expression as in Definition 6. Thus, from condition 4 in the definition, we observe that

$J_3^{\mathcal{O}_3(q)} = \{\pi_1, \ldots, \pi_n\} \subseteq \sigma_{m,n_m}$ for all $m = p + dl + \mathcal{O}_3(q)$, for all $d \in \{0, 1, 2 \ldots\}$ and for all $q \in \{1, 2, \ldots, \ell\}$. Hence, we can conclude that $\sigma \vDash \phi_{rec}$.

Thus, by combining the results from Cases 1, 2, 3 and 4 with conjunction [10] (Fig 5.2, pp 236 [10]), we can conclude that σ satisfies $\phi = \phi_{globe} \wedge \phi_{reach} \wedge \phi_{rec} \wedge \phi_{act}$. That is, $\sigma \vDash \phi$. ■

Next we state Theorem 1 which provides a theoretical guarantee that if Algorithm 2 is feasible, then the trace of the resulting system trajectory satisfies the specification.

Theorem 1. *Given a $LTL_{robotic}$ specification ϕ and a lasso template $\mathcal{O}_2$ and $\mathcal{O}_3$, let $\mathcal{R}_{lasso}$ be the lasso-type constrained reachability sequence as in* (3.5) *generated from Algorithm 1. If Algorithm 2 is feasible, then the trace of the system trajectory $x(t)$ satisfies ϕ.*

Proof. As per Algorithm 2, each Σ_i is encoded as constraint(s) with ZCBFs for all $i \in \{1, 2, \ldots, p + \ell\}$. From Proposition 1, this guarantees forward invariance of the atomic propositions that need to remain true or need to remain false. Since the QP from Algorithm 2 is feasible, conditions 1 and 5 from Definition 6 are satisfied. Since each Γ_i is encoded as constraint(s) with FCBFs for all $i \in \{1, 2, \ldots, p + \ell\}$, from Proposition 2 (Theorem 6 can also be used, as is discussed in Chapter 5) we can guarantee finite time convergence to atomic propositions that need to be reached in finite time. This satisfies conditions 2, 3 and 4 of Definition 6. Thus, all conditions in Definition 6 are satisfied. Since the QP is feasible, we conclude that Algorithm 2 generates a descendant σ of the lasso template.

From Proposition 3, we know that given a lasso template, any descendant σ of the lasso template is such that it satisfies the specification. From the previous analysis, we know that the QP from Algorithm 2 produces a descendant of the lasso template. The mapping ι being bijective and combining Proposition 3 with the previous analysis, we can conclude that QP from Algorithm 2 produces a trace of the trajectory of the system that satisfies the given specification. That is, $\iota^{-1}(\sigma) = \sigma \vDash \phi$. ■

Note that while Algorithm 2 and Theorem 1 assume that the QP (1.6) is feasible, one

can always use the technique of relaxing the QP as is discussed in Chapter 5, for feasibility. In that case, although feasibility of the controller is more likely, Theorem 1 may no longer hold since the relaxation parameters ϵ can be non-zero so that the corresponding atomic propositions are no longer satisfied. However, such a situation is beyond the scope of this thesis.

3.3 Framework Implementation Results

In this section, we provide a case study implemented in the Robotarium multi-robot testbed at Georgia Tech [67]. The Robotarium consists of differential drive mobile robots which can be programmed using either MATLAB or Python.

Consider a team of three robots: one surveillance robot ($\mathtt{R}_3$) and two attack robots ($\mathtt{R}_1$ and $\mathtt{R}_2$). The surveillance robot needs to collect information regarding the position of two targets, and then return back to the base. Once the information has been relayed to the base by the surveillance robot, the attack robots must visit the targets infinitely often. In addition to this, the attack robots must stay connected with each other at all times, and all the robots must avoid a danger zone where they can be attacked.

Let $\mathcal{D} \subset \mathbb{R}^2$ be the workspace for each robot and let $\mathcal{D}\times\mathcal{D}\times\mathcal{D} \subset \mathbb{R}^6$ be the domain of the three robot system with regions $\mathcal{A} = \{A, B, C, O\}$. The dynamics for each agent $i \in \{1, 2, 3\}$ is

$$\begin{bmatrix} \dot{p}_i^1 \\ \dot{p}_i^2 \\ \dot{\phi}_i \end{bmatrix} = \begin{bmatrix} \cos(\phi_i) & 0 \\ \sin(\phi_i) & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v_i \\ \omega_i \end{bmatrix},$$

where $p_i^1 \in \mathbb{R}$ and $p_i^2 \in \mathbb{R}$ represent the position of the robot, $\phi_i \in (-\pi, \pi]$ represents its orientation, $v_i \in \mathbb{R}$ and $\omega_i \in \mathbb{R}$ are the linear and angular velocity inputs to the robot respectively. Denote $x_i = \begin{bmatrix} p_i^1 & p_2^i & \phi_i \end{bmatrix}^T$, and $\bar{x}_i = \begin{bmatrix} p_i^1 & p_2^i \end{bmatrix}^T$. For implementation purposes in the Robotarium and theoretical reasons associated with the unicycle robot as discussed

in [68], we use the NID technique discussed in [69] to control the differential drive robots as a single integrator model. The NID technique allows for control over both input to a differential drive robots as discussed in [68].

Target 1 is labelled as A, target 2 is labelled as B the base is labelled as C, and O is the danger zone (obstacle). The set of atomic propositions is given by $\Pi = \{\pi_i^r, \overline{\pi}_i^r\} \cup \{\pi^{conn}, \overline{\pi}^{conn}\}$ for all $i \in \{1,2,3\}$ and $r \in \{A,B,C,O\}$. The regions *A*, *B*, *C* are defined as $[\![\pi_i^r]\!] = \{x \in \mathcal{D}^3 | h_r(x_i) \geq 0\}$ for all $r \in \{A,B,C,O\}$ and for all $i \in \{1,2,3\}$. For each $[\![\pi_i^r]\!]$ with $i \in \{1,2,3\}$, $r \in \{A,B,C,O\}$, let

$$\pi_i^r = \begin{cases} 1 & \text{if } \bar{x}_i \in r \\ 0 & \text{otherwise.} \end{cases} \tag{3.11}$$

This means $\pi_i^r = 1$ if and only if agent i is in region r. The additional connectivity constraint that must be maintained by $\mathtt{R}_1$ and $\mathtt{R}_2$ is given as $h_{conn}(x) \geq 0$ where

$$h_{conn}(x) = d_{conn}^2(x) - \|\bar{x}_2 - \bar{x}_1\|^2, \tag{3.12}$$

where $d_{conn} : \mathcal{D} \times \mathcal{D} \times \mathcal{D} \to \mathbb{R}$ is the connectivity distance between the two agents that needs to be maintained, and $\|\bar{x}_2 - \bar{x}_1\|$ is the inter-agent distance. We consider

$$d_{conn}^2(x) = (p_2^1 + \delta_1)^2 + \delta_2, \tag{3.13}$$

where δ_1 and δ_2 are constants. The connectivity set corresponding to the proposition π^{conn} is defined as $[\![\pi^{conn}]\!] = \{x \in \mathcal{D}^3 | h_{conn}(x) \geq 0\}$. Such a constraint captures a situation in which the robots have poor connectivity in certain areas of the workspace, which requires them to maintain a closer distance with each other. In areas where the robots have strong connectivity, they are free to maintain a larger distance from each other.

The $LTL_{robotic}$ specification for the task described previously is given by

$$\phi = \left(\Diamond\pi_3^A \wedge \Diamond\pi_3^B \wedge \Diamond\pi_3^C\right) \wedge \Box\Diamond\left(\pi_1^A \wedge \pi_2^B\right) \wedge \Box\Diamond\left(\pi_1^C \wedge \pi_2^C\right) \wedge \Box\left(\pi^{conn} \wedge \neg\pi_1^O \wedge \neg\pi_2^O \wedge \neg\pi_3^O\right). \tag{3.14}$$

From the formalism in Definition 5 and Algorithm 1, we obtain the lasso-type constrained reachability objective,

$$\mathcal{R}_{lasso} = \Big(R_1(\Sigma_1,\Gamma_1)R_2(\Sigma_2,\Gamma_2)R_3(\Sigma_3,\Gamma_3)\Big)\Big(R_4(\Sigma_4,\Gamma_4)R_5(\Sigma_5,\Gamma_5)\Big)^{\omega}$$

where $\Sigma_i = [\![\pi^{conn}]\!] \cap [\![\overline{\pi_1}^O]\!] \cap [\![\overline{\pi_2}^O]\!] \cap [\![\overline{\pi_3}^O]\!]$ for $i = 1,2,3,4,5$, $\Gamma_1 = [\![\pi_3^A]\!]$, $\Gamma_2 = [\![\pi_3^B]\!]$, $\Gamma_3 = [\![\pi_3^C]\!]$, $\Gamma_4 = [\![\pi_1^A]\!] \cap [\![\pi_2^B]\!]$, $\Gamma_5 = [\![\pi_1^C]\!] \cap [\![\pi_2^C]\!]$. Thus, we have five quadratic programs

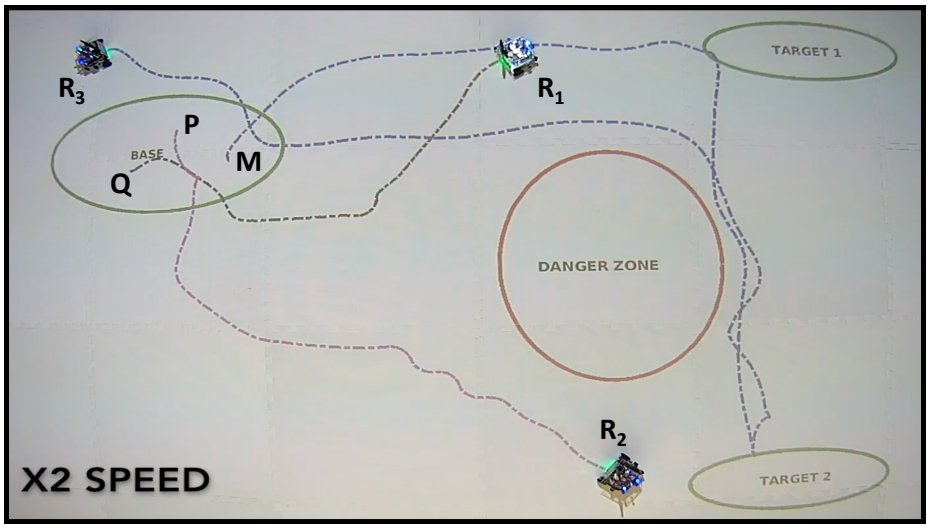


Figure 3.1: A still shot of the trajectories for the robots $\mathtt{R}_1$, $\mathtt{R}_2$ and $\mathtt{R}_3$ for the specification ϕ as in (3.14). Observe that $\mathtt{R}_1$ moves temporarily away from target 1 temporarily in order to satisfy the connectivity constraint dictated by (3.12) and (3.13), but Theorem 6 results in feasible solutions at those points. From the figure, we observe that the robots maintain connectivity and avoid the danger zone at all times.

(corresponding to each reachability objective) that needs to be solved for guaranteed task satisfaction, with the objectives in the suffix part of the lasso sequence solved for infinitely many iterations.

Table 3.1: Average computation time for each quadratic program

Initial Condition of Robots	**QP** 1 **(ms)**	**QP** 2 **(ms)**	**QP** 3 **(ms)**	**QP** 4 **(ms)**	**QP** 5 **(ms)**
Case 1	2.4	3.3	2.9	4.3	2.1
Case 2	2.3	2.1	2.0	4.0	2.0
Case 3	2.6	2.2	2.0	4.1	2.1
Case 4	2.4	2.2	1.8	3.5	2.0
Case 5	2.2	2.0	1.8	3.7	2.0

Next, we use Algorithm 2 to generate the pointwise controller for the system. Each reachability objective $R_i(\Sigma_i, \Gamma_i)$ for all $i \in \{1, 2, 3, 4, 5\}$ is encoded as a QP and is solved sequentially. In particular, if $\mathcal{U}_i(x)$ is the set of feasible control laws that satisfies all the constraints for each reachability objective, then for all $i = \{1, 2, 3, 4, 5\}$ the QP solved is given by,

$$\begin{aligned} \min_{u \in \mathbb{R}^6} \quad & \|u\|_2^2 \\ \text{s.t} \quad & u \in \mathcal{U}_i(x). \end{aligned} \tag{3.15}$$

From Theorem 1 we conclude that these trajectories indeed satisfy the specification ϕ. The switching between the current reachability objective to the next is automatic. It occurs when the state of the system reaches the desired set of states. That is, the switching from reachability objective i to objective $i + 1$ occurs when $x \in \Gamma_i$ for all $i \in \{1, 2, 3, 4\}$.

As is discussed in [68], the computation complexity of strongly convex QPs, a class to which the proposed controller belongs to, is $O((m + d)^3)$ where m is the number of decision variables, and d is the number of constraints in the QP. This is the complexity of most commonly used solvers. We performed the experiment for multiple initial conditions of the robots, and recorded the average computation time for each QP in the lasso sequence. This is summarized in table 3.1. As can be seen, each QP is solved in milliseconds due to the capabilities of modern day QP solvers. This highlights one of the main advantages of our proposed work which is a discretization free approach amenable to real-time implementation.

At all times, $\mathtt{R}_1$ and $\mathtt{R}_2$ stay connected as per the distance dictated by (3.12) and avoid the

danger zone, as seen in Fig 3.1. Thus, we see that by solving this sequence of constrained reachability objectives, the multi-agent system satisfies the specification. Fig 3.1 is a still shot of the experiment conducted on the Robotarium testbed at Georgia Tech [67] [1]

3.4 Smooth Transition Between Quadratic Programs

Suppose the system of interest is an omnidirectional robot with single integrator dynamics $\dot{x} = u$, where $x \in \mathbb{R}^2$ is the state, and $u \in \mathbb{R}^2$ is the control input applied to the robot. Consider a domain $\mathcal{D} \subset \mathbb{R}^2$ containing two regions of interest- region A and region B. Let $\mathcal{A} = \{x \in \mathcal{D} \mid h_A(x) \geq 0\}$ and $\mathcal{B} = \{x \in \mathcal{D} \mid h_B(x) \geq 0\}$, where $h_A : \mathbb{R}^2 \to \mathbb{R}$ and $h_B : \mathbb{R}^2 \to \mathbb{R}$ are continuously differentiable functions. Suppose the task to be satisfied by the robot is to visit region A first followed by region B. Adhering to the methodology suggested in the earlier sections, one can formulate this task as a sequence of two QPs using the following candidate lasso sequence:

$$\mathcal{R}_{\text{lasso}} = \Big(R_1(\varnothing, \Gamma_1) R_2(\varnothing, \Gamma_2)\Big)\Big(R_3(\varnothing, \varnothing)\Big)^{\omega} \tag{3.16}$$

where $\Gamma_1 = \{x \in \mathcal{D} \mid h_A(x) \geq 0\}$ and $\Gamma_2 = \{x \in \mathcal{D} \mid h_B(x) \geq 0\}$. Using our proposed framework, we solve the reachability objectives R_1 and R_2 using two independent quadratic programs described below.

Quadratic Program 1: The first QP is defined to be

$$\begin{aligned} \min_{u \in \mathbb{R}^2} \quad & \|u\|_2^2 \\ \text{s.t} \quad & \frac{\partial h_A(x)}{\partial x} \cdot u \geq -\gamma \cdot \text{sign}(h_A(x)) \cdot |h_A(x)|^{\rho} \end{aligned}$$

where $\gamma > 0$, and $\rho \in [0, 1)$. This QP is solved until the system reaches $\mathcal{A}$.

Quadratic Program 2: Next, the QP constraint is switched to a new one to reflect the

[1] Video of Robotarium experiment - `https://youtu.be/EK1Zxcg-eSE`

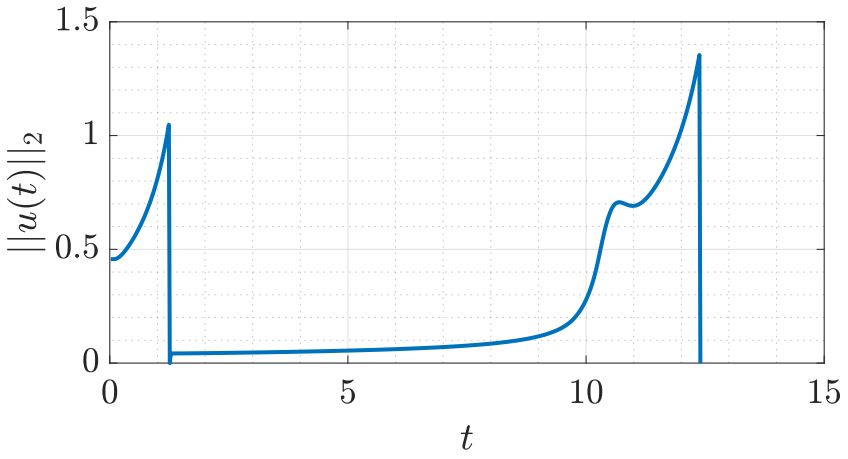


Figure 3.2: The control input generated for the sequence of tasks discussed in Example 1. The sudden switching of the reachability constraint from QP 1 to QP 2 yields a discontinuity in the control law as is seen here. In Section 3.6, we show that using our proposed framework, a continuous control input is obtained.

new task, i.e., convergence to region B. Hence, the second QP is defined to be

$$\begin{aligned} \min_{u \in \mathbb{R}^2} \quad & \|u\|_2^2 \\ \text{s.t} \quad & \frac{\partial h_B(x)}{\partial x} \cdot u \geq -\gamma \cdot \text{sign}(h_B(x)) \cdot |h_B(x)|^\rho \end{aligned}$$

where $\gamma > 0$, and $\rho \in [0, 1)$.

The sudden change in the constraint yields a discontinuous control law as illustrated in Fig 3.2. A similar example of this problem is discussed in [70] and [71]. Hence, discretely switching from one QP to the next creates discontinuities in the control input to the robot, which is undesirable. To address this problem, we propose a new barrier function constraint with time dependent coefficients which guarantee continuity of the control law. Instead of solving a sequence of QPs with discrete transitions, we propose a single QP which smoothly transitions between different barrier function constraints.

To that end, consider a continuous time mobile robotic system in control affine form as in (1.1). We assume that there exists a sequence of tasks to be executed by the system. Below, we provide a formal definition for a task.

Definition 7. *Given a target set $\Gamma \subset \mathcal{D}$ and a safety set $\Sigma \subset \mathcal{D}$, the task $\mathcal{T}_{(\Gamma,\Sigma)}$ is defined as the reachability problem requiring that the system reach Γ in finite time, while staying in Σ for all time.* □

Define $\mathcal{T} = \{\mathcal{T}_{(\Gamma_1,\Sigma)}, \ldots, \mathcal{T}_{(\Gamma_n,\Sigma)}\}$ where each $\mathcal{T}_{(\Gamma_i,\Sigma)}$ is a task that the system must satisfy in finite time and each sequential task is distinct, i.e., $\mathcal{T}_{(\Gamma_i,\Sigma)} \neq \mathcal{T}_{(\Gamma_{i+1},\Sigma)}$ for all i.

Informally, each task $\mathcal{T}_{(\Gamma_i,\Sigma)}$ consists of reachability and invariance constraints. Invariance constraints are assumed to be global constraints which do not change between consecutive tasks; however, the reachability constraints are distinct between tasks. These constraints are encoded in a QP which is solved until the system reaches the target set Γ_i. Suppose there exists m finite-time barrier functions and each target set Γ_i for the task $\mathcal{T}_{(\Gamma_i,\Sigma)}$ is characterized as the intersection of some subset of the m barrier functions. Let $\mathcal{I} = \{1, 2, \ldots, m\}$ be the index set for the finite-time barrier functions. Given a target set Γ_i for a task i, the following definition formalizes the finite-time barrier functions which characterize Γ_i.

Definition 8 (Reachability Set Map)**.** *The reachability set map $\Delta : \{\Gamma_1, \Gamma_2, \ldots, \Gamma_n\} \to 2^{\mathcal{I}}$ yields the finite-time barrier functions characterizing the target set such that $\Gamma_i = \bigcap_{\forall j \in \Delta(\Gamma_i)} \{x \in \mathcal{D} \mid h_j(x) \geq 0\}$ for all $i \in \{1, 2, \ldots, n\}$.* □

Next, we formalize the problem statement addressed in this chapter.

Problem Statement 2. *Given a system in control-affine form as in* (1.1)*, and a sequence of tasks $\mathcal{T} = \{\mathcal{T}_{(\Gamma_1,\Sigma)}, \ldots, \mathcal{T}_{(\Gamma_n,\Sigma)}\}$, synthesize a continuous controller such that the system satisfies each $\mathcal{T}_{(\Gamma_i,\Sigma)}$ within a finite time, while smoothly transitioning between sequential tasks.*

3.5 Barrier Function Based Smooth Task Transition

Smoothly transitioning between tasks requires a transition function which gradually winds down the weight(s) of the constraint(s) corresponding to an accomplished task in the quadratic

program (QP) while ramping up the weight(s) of the constraint(s) of the successive task. In this section, we propose a framework to achieve this.

QP Based Algorithm

Consider a sequence of n reachability tasks represented as $\mathcal{T} = \{\mathcal{T}_{(\Gamma_1,\Sigma)}, \ldots, \mathcal{T}_{(\Gamma_n,\Sigma)}\}$, and m finite time barrier functions indexed by the set $\mathcal{I}$. Introduce transition function $\alpha_i : \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ for all $i \in \mathcal{I}$. Then, for all $t \geq 0$, we formulate a time varying feedback control QP of the form

$$
\begin{aligned}
u^*(t,x) = \min_{u \in \mathbb{R}^m} \quad & \|u\|_2^2 \\
\text{s.t} \quad & \sum_{i=1}^{m} \Big\{ \alpha_i(t)(L_f h_i(x(t)) + L_g h_i(x(t))u(x(t))) \Big\} + \\
& \sum_{i=1}^{m} h_i(x(t)) \frac{\partial \alpha_i(t)}{\partial t} \\
& \geq -\gamma \cdot \tanh\left(-\ln\left(\sum_{i=1}^{m} \exp(-\alpha_i(t) h_i(x(t))) \right) \right) \\
& u \in \mathcal{U}_{\text{safe}}(x) \\
& \|u\|_\infty \leq M,
\end{aligned}
\tag{3.17}
$$

where $\alpha_i(t)$ is determined from Algorithm 4 for all $i \in \mathcal{I}$, and $\gamma > 0$. The first constraint captures the reachability part of each task specification, whereas the second constraint represents the safety requirements to be satisfied by the system. The last constraint represents actuator limits where $M > 0$. This optimization problem must be solved point-wise in time i.e. in a sampled-data fashion.

We construct Algorithm 3 and Algorithm 4, which use (3.17) to generate the control inputs required to satisfy the given sequence of reachability tasks for the system. Algorithm 3 utilizes the function characterized in Algorithm 4. In particular, Algorithm 3 is executed for all $t \geq 0$. The transition functions $\alpha_i(t)$ for all $i \in \mathcal{I}$ and for all $t \geq 0$ are chosen as per Algorithm 4 point-wise in time. The functions $\kappa^{\uparrow} : \mathbb{R}_{\geq 0} \to \mathbb{R}$ and $\kappa^{\downarrow} : \mathbb{R}_{\geq 0} \to \mathbb{R}$ are strictly

increasing and decreasing continuously differentiable functions, respectively. Assuming T_i is the time instant at which the robot reaches target set Γ_i, the functions $\kappa^{\uparrow}$ and $\kappa^{\downarrow}$ are chosen such that $\kappa^{\uparrow}(t - T_i) \mid_{t=T_i} = 0$ and $\kappa^{\downarrow}(t - T_i) \mid_{t=T_i} = 1$. Choices for $\kappa^{\uparrow}$ and $\kappa^{\downarrow}$ include functions such as sine, cosine, hyperbolic tangent, sigmoid, etc.

Algorithm 3 Smooth Transition Between Sequential Reachability Tasks

Input : h_i **for all** $i \in \mathcal{I}$, $\gamma > 0$

for $i \in \{1, \ldots, n\}$ **do**
  $T_i \leftarrow 0$
  $s \leftarrow 0$
  while $x \notin \Gamma_i$ **do**
    Compute-$\alpha(t, T_i, x, i, s)$
    Solve the QP (3.17)
    Apply $u(t, x)$ to the system
  end while
  $T_i \leftarrow t$
  $s \leftarrow 1$
  while $\alpha_k < 1 \ \ \& \ \ \alpha_j > 0 \ \ \forall \ k \in \Delta(\Gamma_{i+1}), j \in \Delta(\Gamma_i)$ **do**
    Compute-$\alpha(t, T_i, x, i, s)$
    Solve the QP (3.17)
    Apply $u(t, x)$ to the system
  end while
end for

Algorithm 4 Compute Transition Function α

Input : t, T_i, x, i, s

if $s = 0$ **then**
  $\alpha_j \leftarrow 1$ for all $j \in \Delta(\Gamma_i)$
  $\alpha_k \leftarrow 0$ for all $k \in \mathcal{I} \setminus \Delta(\Gamma_i)$
else if $s = 1$ **then**
  $\alpha_j \leftarrow \kappa^{\downarrow}(t - T_i)$ for all $j \in \Delta(\Gamma_i)$
  $\alpha_l \leftarrow \kappa^{\uparrow}(t - T_i)$ for all $l \in \Delta(\Gamma_{i+1})$
end if

3.5.1 Continuity of QP-based Controller & Reachability Task Satisfaction Guarantee

The following theorem provides a composite barrier function constraint which allows for smooth task transition.

Lemma 1. *Consider a task $\mathcal{T}_{(\Gamma,\Sigma)}$ with the target set Γ defined as $\Gamma = \bigcap_{i\in\mathcal{P}} \{x \in \mathcal{D} \mid h_i(x) \geq 0\}$ where $\mathcal{P} = \{1, 2, \ldots, k\}$, with each barrier function h_i bounded. That is, $h_i(x) \leq M_i$ for all $x \in \mathcal{D}$ where $M_i > 0$. If there exists a continuous controller $u : \mathcal{D} \to \mathbb{R}^m$ such that for all $x \in \mathcal{D}$ and for all $t \geq 0$,*

$$\sum_{i\in\mathcal{P}} \Big\{ (L_f h_i(x(t)) + L_g h_i(x(t)) u(x(t))) \Big\} \geq -\gamma \cdot \tanh\Bigg(-\ln\Bigg(\sum_{i\in\mathcal{P}} \exp(-h_i(x(t))) \Bigg) \Bigg) \tag{3.18}$$

then there exists a time instance $0 < T < \infty$ such that $x(T) \in \Gamma$.

Proof. By contradiction, suppose for some $x(0) \in \mathcal{D}\backslash\Gamma$, the control law that satisfies (3.18) is such that there does not exist a time $0 < T < \infty$ so that $x(T) \in \Gamma$. Hence we have $\min_{i\in\mathcal{P}}(\{h_i(x(t))\}) < 0$. From [72], p. 72, we therefore have

$$-\ln\Bigg(\sum_{i\in\mathcal{P}} \exp(-h_i(x(t))) \Bigg) < 0$$

for all $t \geq 0$. Since $\tanh(x) < 0$ for all $x < 0$, we have for all $t \geq 0$

$$\tanh\Bigg(-\ln\Bigg(\sum_{i\in\mathcal{P}} \exp(-h_i(x(t))) \Bigg) \Bigg) = -\beta < 0$$

Observe that he inequality (3.18) can be rewritten as

$$\frac{d}{dt} \sum_{i\in\mathcal{P}} h_i(x(t)) \geq \gamma \cdot \beta \tag{3.19}$$

By integration of (3.19) using the fundamental theorem of calculus and from (3.18), we

thus get

$$\sum_{i\in\mathcal{P}} h_i(x(t)) \geq \gamma \cdot \beta \cdot t + \sum_{i\in\mathcal{P}} h_i(x(0))$$

Observe that as $t \to \infty$, $\sum_{i\in\mathcal{P}} h_i(x(t)) \to \infty$. However, this is a contradiction since the barrier functions are bounded, and hence we have $\sum_{i\in\mathcal{P}} h_i(x(t)) < \sum_{i\in\mathcal{P}} M_i$ for all $t \geq 0$. Hence there exists a $0 < T < \infty$ such that $x(T) \in \Gamma$. ■

The following theorem reformulates the constraint (3.18) taking into account the transition periods of the transition functions given a sequence of n tasks to be executed by the system.

Theorem 2. *Consider m bounded finite-time barrier functions, i.e., $h_i(x) \leq M_i$ for all $x \in \mathcal{D}$ and where $M_i > 0$ for all $i \in \{1, 2, \ldots, m\}$. Given a sequence of tasks $\mathcal{T} := \{\mathcal{T}_{(\Gamma_1,\Sigma)}, \ldots, \mathcal{T}_{(\Gamma_n,\Sigma)}\}$ with the corresponding transition functions $\alpha_j(t)$ chosen according to Algorithm 4 for all $j \in \Delta(\Gamma_i)$, for all $i \in \{1, \ldots, n\}$, and for all $t \geq 0$, if $u : \mathbb{R}_{\geq 0} \times \mathcal{D} \to \mathbb{R}^m$ is a continuous controller such that for all $x \in \mathcal{D}$ and $t \geq 0$, we have*

$$\sum_{i=1}^{m} \left\{ \alpha_i(t)(L_f h_i(x(t)) + L_g h_i(x(t))u(t, x(t))) \right\} + \sum_{i=1}^{m} h_i(x(t)) \frac{\partial \alpha_i(t)}{\partial t} \geq -\gamma \cdot \tanh\left(-\ln\left(\sum_{i=1}^{m} \exp(-\alpha_i(t) h_i(x(t))) \right) \right), \quad (3.20)$$

then there exists a sequence of finite time instances $0 < T_1 < T_2 < \ldots < T_n < \infty$ such that $x(T_i) \in \Gamma_i$ for all $i \in \{1, \ldots, n\}$ i.e. the task sequence $\mathcal{T}$ is solved.

Proof. The proof is analogous to the proof of Lemma 1. Consider the first task $\mathcal{T}_{(\Gamma_1,\Sigma_1)}$. From Lemma 1 we know that there exists a finite time $0 < T_1 < \infty$ such that $x(T_1) \in \Gamma_1$. Suppose by contradiction, for some $x(T_1) \in \mathcal{D} \backslash \Gamma_2$, the constraint (3.20) is satisfied for all $t \geq T_1$ but there does not exist a finite time $0 < T_2 < \infty$ such that $x(T_2) \in \Gamma_2$. Hence we have that $\min_{i\in\{1,2,\ldots,m\}} (\{\alpha_i(t) h_i(x(t))\}) = \min_{i\in\Delta(\Gamma_2)} (\{\alpha_i(t) h_i(x(t))\}) < 0$. Following a proof

methodology similar to Lemma 1, we can prove that there exists a finite time $0 < T_2 < \infty$ such that $x(T_2) \in \Gamma_2$. Following a successive proof by induction for tasks 3 to n, we can thus conclude that there exists a sequence of finite time instances $0 < T_1 < T_2 < \ldots < T_n < \infty$ such that $x(T_i) \in \Gamma_i$ for all $i \in \{1, 2, \ldots, n\}$ i.e. the task sequence $\mathcal{T}$ is solved. ■

Since we are interested in guaranteeing continuity of the controller, the following theorem proves that the proposed controller (3.17) as used in Algorithm 3 is continuous.

Theorem 3. *If Algorithm 1 is feasible for all $t \geq 0$, then the control input, computed by solving* (3.17)*, applied to the system is continuous.*

Proof. Consider an indicator variable for the time, θ with $\dot{\theta} = 1$. Considering the new state of the system as $\hat{x} = \begin{bmatrix} x & \theta \end{bmatrix}^T$, the constraint (3.20) can be reformulated as

$$\sum_{i=1}^{m} \left\{ \alpha_i(\hat{x})(L_f h_i(\hat{x}) + L_g h_i(\hat{x}) u(\hat{x})) \right\} + \sum_{i=1}^{m} h_i(\hat{x}) \frac{\partial \alpha_i(\hat{x})}{\partial \theta} + \gamma \cdot \tanh\left(- \ln\left(\sum_{i=1}^{m} \exp(-\alpha_i(\hat{x}) h_i(\hat{x})) \right) \right) \geq 0.$$

The reformulated inequality is now the barrier function constraint (3.18) for the new time invariant system with the augmented state $\hat{x}$. Similarly, the constraint $u(x) \in \mathcal{U}_{\text{safe}}(x)$ can be reformulated in terms of the new state. These constraints are quasi-convex in the control u for all $\hat{x} \in \mathbb{R}^{n+1}$ and the cost is quasi-convex. In addition, the input is constrained over a compact set (third constraint in (3.17)). However, since $\alpha_i(\hat{x})$ for all $i \in \mathcal{I}$ is chosen point-wise in time as per Algorithm 4, the continuity of the controller must be established by analyzing the transition phase, reachability phase, and the time instant at which the transition between phases occurs as per Algorithm 3. Since all the assumptions of Proposition 8 in [73] are satisfied, the controller computed from Algorithm 3 is continuous for the entirety of the transition and the reachability phase. Since the functions $\kappa\uparrow$ and $\kappa\downarrow$ are chosen such that, $\kappa\uparrow(t - T_i)\,|_{t=T_i} = 0$ and $\kappa\downarrow(t - T_i)\,|_{t=T_i} = 1$, the constraint (3.20) does not suffer from discontinuities when switching between the reachability and the transition phase. Thus, the

controller computed as per Algorithm 3 is continuous for all $t \geq 0$. ■

3.6 Smooth Task Transition Implementation Results

In this section, we present the implementation[2] of the theoretical framework of smooth barrier function transitions, in the Georgia Tech Robotarium [67].

Consider a differential drive mobile robot with dynamics

$$\dot{x} = v \cdot cos(\phi)$$

$$\dot{y} = v \cdot sin(\phi)$$

$$\dot{\phi} = \omega$$

where $x \in \mathbb{R}$ and $y \in \mathbb{R}$ are the position coordinates of the robot, $\phi \in [-\pi, \pi)$ is the orientation, $v \in \mathbb{R}$ is the linear velocity input, and ω is the angular velocity input. Denote $\tilde{x} = \begin{bmatrix} x & y & \phi \end{bmatrix}^T$, $\hat{x} = \begin{bmatrix} x & y \end{bmatrix}^T$, and $\mathcal{D} \subset \mathbb{R}^3$. Then we have $f(\tilde{x}) = 0$ and $g(\tilde{x}) = \begin{bmatrix} \cos(\phi) & 0 \\ \sin(\phi) & 0 \\ 0 & 1 \end{bmatrix}$. Let $u = \begin{bmatrix} v & \omega \end{bmatrix}^T$ be the input vector to the robot. The experiment was conducted on the Robotarium multi-robot testbed at Georgia Tech. For more details on the hardware specifications and platform setup, please refer [67]. Since the differential drive robot model is non-holonomic which leads to controllability issues, we use a technique known as the Near Identity Diffeomorphism (NID) to control the robot, as detailed in [67].

The workspace consists of four regions of interest - three goal regions (A, B and C) and an obstacle O. Fig. 3.3 illustrates the domain of interest. Here, the robot must initially visit region A, followed by region B and ultimately region C while avoiding the obstacle. The sequence of tasks is given by $\mathcal{T} = \{\mathcal{T}_{(\Gamma_A, \Sigma_O)}, \mathcal{T}_{(\Gamma_B, \Sigma_O)}, \mathcal{T}_{(\Gamma_C, \Sigma_O)}\}$, where $\Gamma_A = \{\tilde{x} \in \mathcal{D} \mid h_1(\tilde{x}) \geq 0\}$, $\Gamma_B = \{\tilde{x} \in \mathcal{D} \mid h_2(\tilde{x}) \geq 0\}$, $\Gamma_C = \{\tilde{x} \in \mathcal{D} \mid h_3(\tilde{x}) \geq 0\}$, and $\Sigma_O = \{\tilde{x} \in \mathcal{D} \mid$

[2] https://github.com/gtfactslab/2020_ContinuousTaskBarriers

$h_4(\tilde{x}) \geq 0\}$, where each $h_i : \mathcal{D} \to \mathbb{R}$ for all $i \in \{1,2,3,4\}$ is a continuously differentiable function. For regions A, B and C, the functions are given by $h_1(\tilde{x}) = 1 - (\hat{x} - C_A)^T P_A(\hat{x} - C_A)$, $h_2(\tilde{x}) = 1 - (\hat{x} - C_B)^T P_B(\hat{x} - C_B)$ and $h_3(\tilde{x}) = 1 - (\hat{x} - C_C)^T P_C(\hat{x} - C_C)$ where P_A, P_B, and P_C are diagonal matrices with the inverses of the dimensions of the ellipsoids as the non-zero entries, and C_A, C_B and C_C are the centers of each ellipsoidal region. The obstacle is modeled using a weighted polar L_p function as described in [74] with parameters $p = 6$, $\sigma = (0.7, 0.2)$, $\theta_\kappa = \frac{\pi}{2}$, $c = 1$ and $\theta_0 = \text{sign}(\kappa) \cdot \frac{\pi}{2}$.

At each time step t, Algorithm 1 is executed and the control input is applied to the robot. The QP that is solved to execute the entire sequence of tasks is

$$
\begin{aligned}
u^*(t,\tilde{x}) = & \min_{u \in \mathbb{R}^2} \quad \|u\|_2^2 \\
& \text{s.t} \sum_{i=1}^{3} \left\{ \frac{d\alpha_i(t) h_i(\tilde{x}(t))}{dt} \right\} \\
& \geq -\gamma \cdot \tanh\left(-\ln\left(\sum_{i=1}^{3} \exp(-\alpha_i(t) h_i(\tilde{x}(t))) \right) \right) \\
& L_f h_4(\tilde{x}(t)) + L_g h_4(\tilde{x}(t)) u(t,\tilde{x}(t)) \geq -\gamma h_4(\tilde{x}(t))^3 \\
& \|u\|_\infty \leq 10
\end{aligned}
$$

where $L_f h_i(\tilde{x}(t)) = \frac{\partial h_i(\tilde{x}(t))}{\partial \tilde{x}} f(\tilde{x}) = 0$ and $L_g h_i(\tilde{x}(t)) = \frac{\partial h_i(\tilde{x}(t))}{\partial \tilde{x}} g(\tilde{x})$ for all $i \in \{1,2,3,4\}$. The average time to solve the QP was between 3ms to 7ms. The increasing and decreasing functions in Algorithm 4 are chosen as $\kappa^{\uparrow} = \sin^2(t - T_i)$ and $\kappa^{\downarrow}(t) = \cos^2(t - T_i)$ respectively, where T_i is the recorded time instant at which the robot reaches region i. Fig. 3.4 demonstrates the continuous change in the solution space as the robot is executing the QP. Observe that the discontinuous switching in the control input is avoided due to the modified constraint in (3.20). This is in contrast to the discontinuities shown in Fig 3.2 where the traditional discrete QP switching method was used. A video of the implementation is provided.[3]

[3]Video of the implementation- `https://youtu.be/eKhXiJkQH8w`

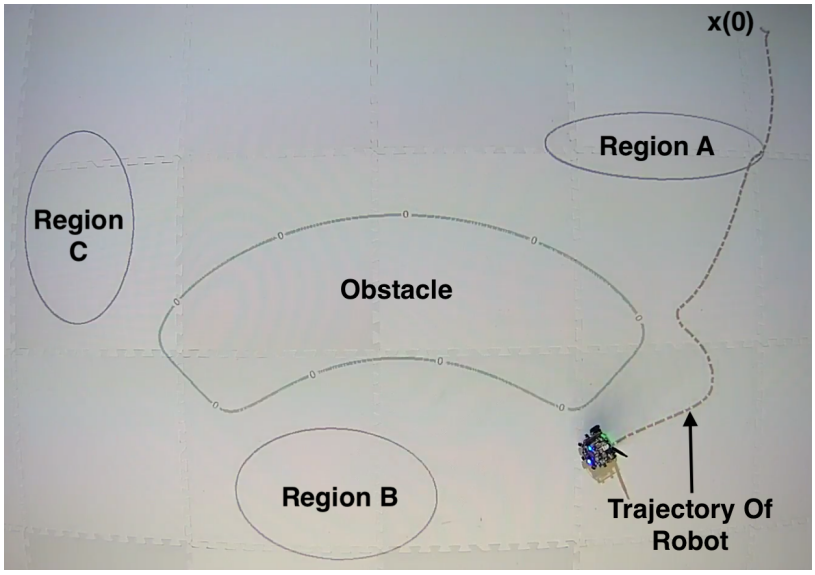


Figure 3.3: A still shot of the implementation of Algorithm 3 conducted on the Robotarium. The robot first visits region A, followed by region B, and lastly region C, while avoiding the obstacle.

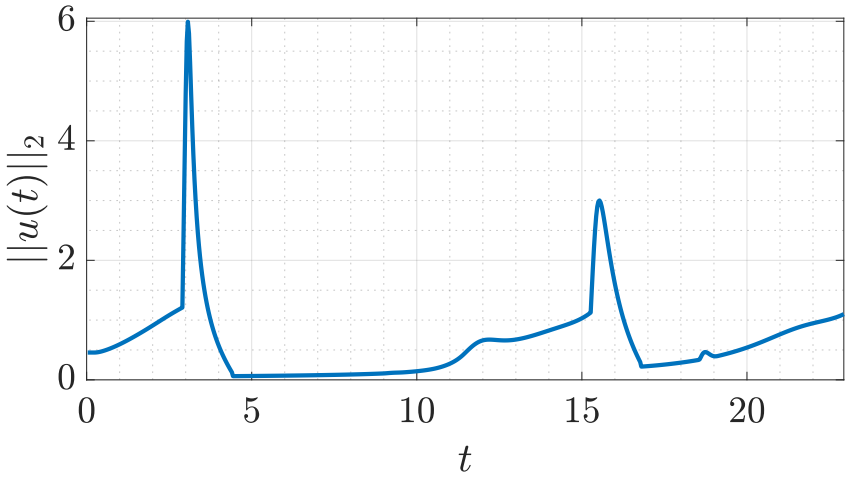


Figure 3.4: The control input u generated in the Robotarium simulator for the task specification as discussed in the experimental setup. Observe that the control generated by our framework is continuous, which is in contrast to Fig 3.2.

3.7 Concluding Remarks

In this chapter, a framework for translating a given used defined specification into a sequence of barrier-certificate based controllers was discussed. We provided algorithms that automatically convert a given task in LTL to a sequence of reachability problems encoded as quadratic programs with the barrier functions as constraints. We provided a formal guarantee that solving such a sequence of reachability objectives produces a system trajectory that satisfies the given specification.

We also proposed a barrier function-based method to ensure smooth transition between different reachability tasks in the lasso sequence. A new composite barrier function constraint was introduced by endowing individual barrier functions with time varying transition functions, which allow for smooth transitions between different objectives. In order to facilitate real-time implementation capabilities, we proposed an algorithm which incorporates the proposed barrier function constraint. Lastly, we proved that the proposed controller is continuous. We provided robotic implementation results for both the synthesis framework, and the method to smoothly transition between different objectives.

Next, we describe a complex multi-robot scenario executed using barrier function based controllers. We discuss the solution methodology, and discuss some of the key assumptions in these approaches, and provide solutions for the same in the remainder of this thesis.

CHAPTER 4
SECURING A BUILDING: OBSERVATIONS FROM A COMPLEX MULTI-ROBOT SCENARIO

In this chapter, a complex multi-robot scenario, where behaviors of the robots are characterized using barrier function-based constraints, is detailed. There are two objective of this chapter: First is to illustrate the level of complex task specifications that can be accomplished using barrier certificate-based controllers, and secondly to show some of the key assumptions that are prevalent in barrier function-based techniques which must be addressed when using such controllers for safety-critical functions. This chapter will serve as a technical motivation for the remainder of the work in this thesis.

Introduction

From a motion controls perspective, one notable requirement is given by the need to define actions capable to solve team-wise objectives on the basis of locally available information. For instance, different extensions of the consensus equation have been used to arrive at locally defined controllers with provable, global convergence properties [75]. In this way, it is possible to construct coordinated controllers for the solution of motion control problems, such as rendezvous [76] [77], cyclic pursuit [78], formation control [79] [80], coverage [81] [82], leader-based control [83], and flocking [84].

For the correct execution of the controllers mentioned, a sufficiently rich set of information needs to be available to the robots. Representing the flow of information between the robots through *graphs*, with vertices and edges being respectively the robots and the pair-wise ability of sharing information, those conditions can be encoded in terms of particular graphs that need to exist between the robots. For example, rendezvous requires a spanning out-branching tree [83], cyclic-pursuit requires a cyclic graph [78], formation

control a rigid graph [83], and a Delaunay graph is required for most of coverage control problems [81].

Even though the coordinated behaviors mentioned above can address a number of different tasks, they have limited utility in the context of real-world missions, which can rarely be represented as single tasks. However, the utility of these behaviors can be greatly expanded if they are sequenced together. But, for a construction like this to work, it is necessary that the required information is available to the robots as they transition from one behavior to the next.

As such, the problem of composing different behaviors, can be recast in terms of the ability of the robots to establish the interactions needed at each stage of a mission. In particular, when the communication between agents depends on their relative configurations (e.g. relative distance or orientation), realizing a certain communication structure directly affects the configuration of the system, which in turn, affects the execution of the mission itself. In order to overcome this coupling, we separate the problem of generating a sequence of behaviors that corresponds to the solution of a mission objective (e.g. [85]) from their composition. In this work we focus on the problem of designing a composition framework given a sequence of coordinated behaviors. Although the focus of this chapter is confined to motion control tasks, our framework is applicable to other forms of autonomous collaboration where desired interaction structures between the robots are required by the mission, e.g., sharing of resources in heterogeneous teams [86] or coordinated manipulation [87].

The contribution of this chapter are twofold. Firstly, extending the results in [9], we propose a fully decentralized framework for composing a given sequence of multi-robot coordinated behaviors. Secondly, responding to the lack of established large-scale scenarios for the testing of multi-robot techniques, we propose a scenario called *Securing a Building*, which is rich and complex enough to capture many challenges and objectives of real-world implementations. The significance of our framework is demonstrated through implementation of the Securing a Building scenario on a team of mobile robots.

4.1 Problem Statement

We denote the state of a team of n homogeneous mobile robots operating in a d-dimensional and connected domain $\mathcal{D}$ as $x(t) = [x_1(t)^T, \ldots, x_n(t)^T]^T \in \mathcal{D} \subset \mathbb{R}^{dn}$ where $x_i(t) \in \mathbb{R}^d$ is the position of robot i at time t. As part of the coordinated nature of the behaviors being performed by the robots, each robot executes a control protocol which depends on the state of the subset of robots with which it interacts. We assume robots can communicate if the distance between them is less or equal to a sensing threshold $\Delta \in \mathbb{R}_{>0}$. Thus, the list of possible interactions between agents are described by a time-varying, undirected, proximity graph $\mathcal{G}(t) = (V, E(t))$, where $V = \{1, \ldots, n\}$ is the set of nodes representing the robots and $E(t)$ is the set of interacting pairs at time t, where

$$E(t) = \{(i, j) \in V \times V \,|\, \|x_i(t) - x_j(t)\| \leq \Delta\}. \tag{4.1}$$

For each robot $i = 1, \ldots, n$, we denote the set of available neighbors at time t as $\mathcal{N}_i(t) = \{j \in V \,|\, (i, j) \in E(t)\}$, which depends on the position of the robots at time t.

The ensemble dynamics of the multi-agent system is described by

$$\dot{x} = f(x) + g(x)\, u \tag{4.2}$$

where f and g are continuous locally Lipschitz continuous functions and $u = [u_1^T, \ldots, u_n^T]^T \in U \subset \mathbb{R}^m$ is the vector of inputs, which depends on the particular behavior being executed. At all times, the control input u in (4.2) is given by a controller $\mathcal{U}$, which can be defined as a state feedback law $\mathcal{U} : \mathcal{D} \mapsto U$ or by a combination of both external parameters and state feedback law $\mathcal{U} : \mathcal{D} \times \Theta \mapsto U$, where Θ is a space of parameters appropriate for the behavior. For instance, the controller corresponding to a *weighted consensus* belongs to the first case. On the other side, a leader-follower protocol where followers maintain prescribed inter-agent distances is described by a controller that depends on both state feedback (followers'

control) and exogenous parameters (leader's goal) (see Section 4.3 for examples).

We represent a mission by an ordered sequence of M coordinated behaviors

$$\pi = \{\mathcal{B}_1, \ldots, \mathcal{B}_M\}. \tag{4.3}$$

The k^{th} behavior in π is defined by the pair

$$\mathcal{B}_k = \{\mathcal{U}_k, \mathcal{G}_k\}, \tag{4.4}$$

where $\mathcal{U}_k$ represents the coordinated controller described above and $\mathcal{G}_k$ is the interaction graph required by behavior $\mathcal{B}_k$ to function properly. We assume the list of behaviors π to be fixed and available to all robots. We will use the term *behavior* to refer to a generalized multi-robot controller in the form (4.4) and to *task* as the objective of the controller.

Each behavior requires a certain interaction structure between the robots (i.e., pairs of robots that need to be neighbors). With reference to (4.4), we describe an interaction structure via the graph $\mathcal{G}_k = (V, E_k)$. Thus, denoting by $t_k^{\vdash}$ and $t_k^{\dashv}$ the starting and ending times for behavior k, the robots' configuration needs to satisfy $\mathcal{G}_k \subseteq \mathcal{G}(t)$ for all $t \in [t_k^{\vdash}, t_k^{\dashv}]$. In other words, as shown in Fig. 4.1, the interaction structure required by each behavior needs to be a spanning graph of the graph induced by the state of the agents during the interval of time the behavior is executed. At this point, given a list of behaviors constituting the mission π and the corresponding multi-robot controllers, we want to design a procedure that enables robots to assemble and maintain the communication graph required by each behavior.

Problem Statement 3. *Given an ordered sequence of coordinated behaviors denoted by* $\pi = \{\mathcal{B}_1, \ldots, \mathcal{B}_M\}$*, where each* $\mathcal{B}_k = \{\mathcal{U}_k, \mathcal{G}_k\}$ *can be completed by the robots in finite-time,*

design a feedback control policy to compose the behaviors such that

$$\mathcal{G}(t) \supseteq \begin{cases} \mathcal{G}_k & t \in [t_k^{\vdash}, t_k^{\dashv}] \\ \mathcal{G}_k \cup \mathcal{G}_{k+1} & t \in (t_k^{\dashv}, t_{k+1}^{\vdash}) \end{cases} \quad \forall\, k = 1, \ldots, M-1. \tag{4.5}$$

4.2 Composition of Coordinated Behaviors

In addition to the list π, transitions between behaviors need to be synchronized, i.e., for each behavior $\mathcal{B}_k$, $k = 1, \ldots, M$, robots must 1) start assembling $\mathcal{G}_{k+1}$ only after all robots have completed $\mathcal{B}_k$ and 2) start executing $\mathcal{B}_{k+1}$ only after condition $\mathcal{G}_{k+1} \subseteq \mathcal{G}(t)$ is satisfied. We assume the existence of a discrete counter $\sigma \in [1, \ldots, M]$ which indicates the active behavior and a binary signal

$$\eta(\sigma) = \begin{cases} 1 & \text{if} \quad \mathcal{G}_k \subseteq \mathcal{G}(t) \\ 0 & \text{o.w.} \end{cases} \tag{4.6}$$

which describes whether the interaction structure required by behavior $\mathcal{B}_\sigma$ is available. In this section, we assume both signals to be controlled by a supervisor and made available to the robots at all times, e.g., through a dedicated static communication network. In the next section, we discuss the extension to a fully distributed framework.

Following from the communication modality assumed for the robots, communication constraints can be expressed in terms of relative distance between the robots. In other words, behavior $\mathcal{B}_k$ can be correctly executed if, for all $t \in [t_k^{\vdash}, t_k^{\dashv}]$, all the distances between pairs in E_k are below the proximity threshold Δ. To this end, a convenient pair-wise connectivity FCBF can be defined as

$$h_{ij}^c(x) = \Delta^2 - \|x_i - x_j\|^2, \tag{4.7}$$

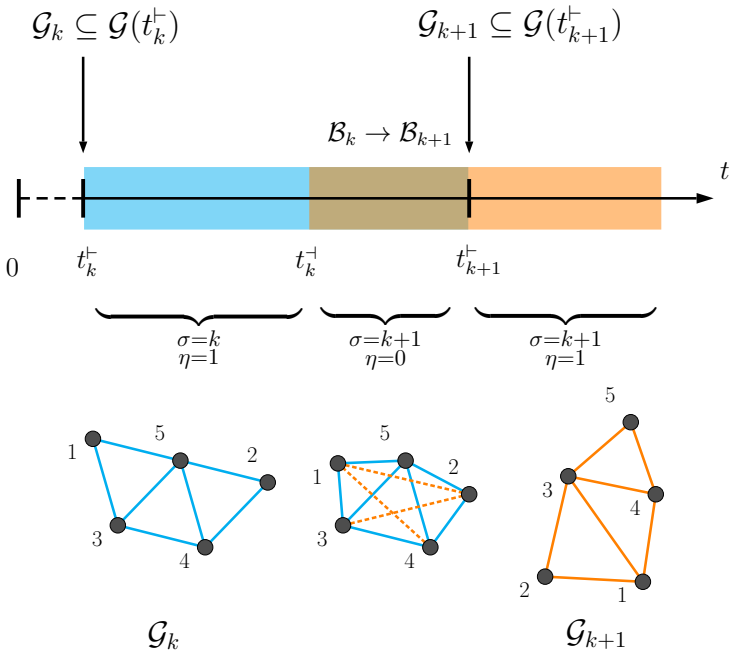


Figure 4.1: Schematic representation of the behaviors sequencing framework. Behavior $\mathcal{B}_k$ is executed during the blue portion of the timeline and $\mathcal{B}_{k+1}$ is executed during the orange portion. Sequential execution of behaviors requires each agent to reach a spatial configuration such that the desired graph is a spanning graph of the communication graph, i.e., $\mathcal{G}_k \subseteq \mathcal{G}(t_k^{\vdash})$ and $\mathcal{G}_{k+1} \subseteq \mathcal{G}(t_{k+1}^{\vdash})$ respectively.

and we note that if $\|x_i - x_j\| \leq \Delta$, then $h_{ij}^c(x) \geq 0$. In addition, the edge-level and ensemble-level connectivity constraint sets for behavior $\mathcal{B}_k$ are

$$\mathcal{C}_{ij}^c = \{x \in \mathcal{D} \,|\, h_{ij}^c(x) \geq 0\} \tag{4.8}$$

$$\mathcal{C}_k^c = \{x \in \mathcal{D} \,|\, h_{ij}^c(x) \geq 0,\ \forall (i,j) \in E_k\}. \tag{4.9}$$

Following the definition given in (1.4), the admissible set of control inputs that guarantees finite-time convergence to $\mathcal{C}_k^c$ is:

$$K_k^c(x) = \{u \in U \,|\, \dot{h}_{ij}^c(x) + \bar{\alpha}_{\rho,\gamma}(h_{ij}^c(x)) \geq 0, \quad \forall (i,j) \in E_k\} \tag{4.10}$$

Theorem 4. *Denoting with x_0 the initial state of the system with dynamics (4.2), any controller $\mathcal{U} : \mathcal{D} \mapsto U$ such that $\mathcal{U}(x_0) \in K_k^c(x_0)$ for all $x_o \in \mathcal{D}$, will drive the system to $\mathcal{C}_k^c$ within time*

$$T_k = \max_{(i,j)\in E_k | h_{ij}^c(x_0)<0} \left\{ \frac{1}{\gamma(1-\rho)} |h_{ij}^c(x_0)|^{1-\rho} \right\}. \tag{4.11}$$

Proof. Consider all pairs of agents i and j, such that $(i,j) \in E_k$. If $h_{ij}^c(x_0) \geq 0$, i.e., agents i and j are within communication distance, the forward invariance property of $\mathcal{U}$, guarantees that i and j will stay connected. In this case, the state will reach $\mathcal{C}_{ij}^c$, within time $T_{ij} = 0$. On the other side, consider $h_{ij}^c(x_0) < 0$. Any $\mathcal{U}(x_0) \in K_k^c(x_0)$ satisfies the finite-time convergence barrier certificates, and because of Proposition 2, if $x_0 \notin \mathcal{C}_{ij}^c$, then $x(T_{ij}) \in \mathcal{C}_{ij}^c$, with

$$T_{ij} \leq \frac{1}{\gamma(1-\rho)} |h_{ij}^c(x_0)|^{1-\rho}. \tag{4.12}$$

Since every communication constraint $\mathcal{C}_{ij}^c$ will be reached within time T_{ij}, the total time required to drive $x(t)$ to $\mathcal{C}_k^c$ is upper bounded by

$$T_k = \max_{(i,j)\in E_k | h_{ij}^c(x_0)<0} T_{ij}. \tag{4.13}$$

■

When selecting control inputs from set (4.10), the system (4.2) will satisfy requirements for behavior $\mathcal{B}_k$ in finite time.

Finite-Time Convergence Control Barrier Functions

Once behavior $\mathcal{B}_{k-1}$ is completed, robots are required to converge to the set $\mathcal{C}_k^c$ before behavior $\mathcal{B}_k$ can start. Under the lead of the external supervisor, the change of behavior is communicated to the robots through the signal σ, which transitions from $k-1$ to k once $\mathcal{B}_{k-1}$ is completed. Now, although finite-time convergence to $\mathcal{C}_k^c$ can be achieved by selecting any control input in $K_k^c(x)$, we seek to minimally perturb the execution of the

behavior just concluded, namely $\mathcal{B}_{k-1}$. This can be accomplished by solving a problem similar to the one proposed in [19], which we adapt to our framework. Denoting with $\hat{u}_k = \mathcal{U}_k(x)$ the nominal control input from behavior $\mathcal{B}_k$, during transition between $\mathcal{B}_{k-1}$ and $\mathcal{B}_k$ the actual control input to the robots u^* is defined as

$$u^* = \arg\min_{u \in U} \|\widehat{u}_{k-1} - u\|^2 \tag{4.14}$$

subject to

$$L_f\, h_{ij}^c + L_g\, h_{ij}^c\, u + \bar{\alpha}_{\rho,\gamma}(h_{ij}^c) \geq 0, \tag{4.15}$$

for all $(i,j) \in E_{k-1} \cup E_k$. Once all required edges E_k are established (i.e., $\eta = 1$), edges in E_{k-1} are no longer necessary. At this point, under the effect of the controller $\mathcal{U}_k$, the list of constraints in (4.15) is substituted with

$$L_f\, h_{ij}^c + L_g\, h_{ij}^c\, u + \bar{\alpha}_{\rho,\gamma}(h_{ij}^c) \geq 0, \tag{4.16}$$

for all $(i,j) \in E_k$. Since the cost function is convex and the inequality constraints (4.15) and (4.16) are control affine, the problem can be solved in real-time. In conclusion, because of the finite-time convergence and forward invariance properties of the above formulation, if $\mathcal{B}_{k-1}$ can be completed and a solution to (4.14-4.15) (or (4.14-4.16)) exists, robots will converge to the configuration required by $\mathcal{B}_k$, and maintain it throughout its execution.

Remark 2. *The solution of (4.14-4.15) (or (4.14-4.16)) is contingent upon the existence of a control input capable to solve all constraints. In other words, $K_k^c(x) \cap K_{k+1}^c(x)$ (or $K_k^c(x)$) should not be empty for all times. For this, it is necessary that a robot's configuration that satisfies all constraints of the problem exists. However, this is not sufficient as the progress towards the desired configuration might be obstructed by constraints on the actuators or deadlock configurations. Although we do not address this directly, it is possible to mitigate feasibility issues by considering, for example, constraints relaxation, sum*

of squares barrier functions, or pre-defined back-up controllers (see [88] and references therein).

Initial Constraints

In addition to the communication constraints considered above, certain missions might require additional conditions to be met before each behavior can start. For example, during the exploration tasks it might be desirable for one robot to always stay within range of communication with a human-operator, or to maintain a minimum distance from an unsafe area. Assuming $\mathcal{B}_k$ requires a number of distinct s_k of such constraints, we encode the entire set of initial conditions through a list of barrier functions $h_\ell^s(x)$, with $\ell = 1, \ldots, s_k$:

$$\mathcal{C}_k^s = \{x \in \mathcal{D} \mid h_\ell^s(x) \geq 0,\ \forall \ell = 1, \ldots, s_k\}. \tag{4.17}$$

Following this definition, we define a set of admissible control inputs similar to the one in (4.10) that will drive the state of the system to the desired set within finite time:

$$K_k^s(x) = \{u \in U \,|\, \dot{h}_\ell^s(x) + \bar{\alpha}_{\rho,\gamma}(h_\ell^s(x)) \geq 0\,, \forall \ell = 1, \ldots, s_k\}. \tag{4.18}$$

The set of controls satisfying both communication and initial conditions constraints can thus be obtained by intersection of set (4.18) and (4.10):

$$K_k(x) = K_k^c(x) \bigcap K_k^s(x). \tag{4.19}$$

We note that the results in Theorem 4 and the formulation of minimally invasive controller in (4.14) still holds valid by considering the set $K_k(x)$ instead of $K_k^c(x)$ as the set of admissible control inputs.

4.3 Distributed Composition of Behaviors

The composition framework discussed in the previous section reduces to the team-wise minimum norm controller (4.14), which is not directly solvable by individual robots. In addition to this, a centralized supervisor is needed in order to synchronize behavior transitions. In this section, we formulate a decentralized solution to problem 3 which can be implemented by the robots using only information from their neighbors. Furthermore, we also include those additional constraints necessary for the safe operations of the robots, e.g., inter-agent collisions and obstacles avoidance [89]. The formulation is derived following the approach described in [90], which we adapt here to our framework.

4.3.1 Distributed Finite-Time Convergence Control Barrier Functions

The limitation in solving problem (4.14) in a distributed fashion is represented by the fact that knowledge of dynamics, input $\hat{u}$, and state x for the entire team need to be available. In addition, solution of (4.14), provides the control inputs for the entire team, which are unnecessary to the individual robots.

In order to develop the correct decentralized formulation of (4.14), we first define a decomposition of the dynamics (4.2). We denote by $\mathcal{D}_i \subset \mathbb{R}^d$ and $U_i \subset \mathbb{R}^m$ configuration space and set of feasible controls for agent i respectively. In addition, by denoting with $\bar{f}, \bar{g} : \mathcal{D}_i \mapsto \mathbb{R}^d$ the node-level terms of the control affine dynamics of agent i, the ensemble dynamics can be written as:

$$\dot{x} = \bar{f}(x_i) \otimes \mathbf{1}_n + \left(\bar{g}(x_i) \otimes I_n\right) \begin{bmatrix} u_1 \\ \vdots \\ u_n \end{bmatrix}, \tag{4.20}$$

where $u_i \in U_i$ is the i^{th} robot's control input, $\otimes$ is the Kronecker product, and $\mathbf{1}_n$ and I_n are vector of ones and identity matrix of size n respectively.

Let's consider two sequential behaviors $\mathcal{B}_{k-1}$ and $\mathcal{B}_k$. Upon completion of $\mathcal{B}_{k-1}$, for all edges $(i,j) \in E_k$, robots' configuration should satisfy

$$\dot{h}^c_{ij}(x_i, x_j) + \bar{\alpha}_{\rho,\gamma}(h^c_{ij}(x_i, x_j)) \geq 0. \tag{4.21}$$

From the i^{th} robot's point of view, the set of constraints that need to be satisfied in order to execute the new behavior are

$$\dot{h}^c_{ij}(x_i, x_j) + \bar{\alpha}_{\rho,\gamma}(h^c_{ij}(x_i, x_j)) \geq 0 \quad , \forall j \in \mathcal{N}^i_k, \tag{4.22}$$

where we recall that $\mathcal{N}^i_k$ is the set of neighbors to robot i required by behavior $\mathcal{B}_k$. However, since constraint (4.22) appears exactly twice across the team of robots, it can be relaxed by considering the admissible set of control inputs

$$K^{c,i}_k = \bigcap_{j \in \mathcal{N}^i_k} K^{c,i}_{k,ij} \tag{4.23}$$

with

$$K^{c,i}_{k,ij} = \{u_i \in U_i \,|\, L_{\bar{f}} h^c_{ij} + L_{\bar{g}} h^c_{ij} u_i + \frac{\bar{\alpha}_{\rho,\gamma}(h^c_{ij})}{2} \geq 0\}, \tag{4.24}$$

where dependence from x_i and x_j is omitted for clarity.

Theorem 5. *Denoting with $x_0 = [x^T_{0,1}, \ldots, x^T_{0,n}]^T$ the initial state of a multi-agent system with dynamics described as in (4.20), any controller $\mathcal{U}_i : \mathcal{D}_i^{|\mathcal{N}^i_k|} \mapsto U_i$ such that $\mathcal{U}_i(x_0) \in K^{c,i}_k$ for all $x_0 \in \mathcal{D}_i^{|\mathcal{N}^i_k|}$, will drive the ensemble state to $\mathcal{C}^c_k$ within time*

$$T_k = \max_{\substack{(i,j)\in E_k \\ \text{s.t. } h^c_{ij}(x_{0,i},x_{0,j})<0}} \left\{ \frac{1}{\gamma(1-\rho)} |h^c_{ij}(x_{0,i}, x_{0,j})|^{1-\rho} \right\}. \tag{4.25}$$

Proof. From Proposition 2, agents i and j, with $(i,j) \in E_k$, will satisfy $h^c_{ij} \geq 0$ in finite

time if

$$\dot{h}_{ij}^{c} + \bar{\alpha}_{\rho,\gamma}(h_{ij}^{c}) \geq 0. \tag{4.26}$$

Considering the node level dynamics in (4.20), the constraint (4.26) reduces to

$$\begin{aligned} &\frac{\partial h_{ij}^{c}}{\partial x_i}\left(\bar{f} + \bar{g}u_i\right) + \frac{\partial h_{ij}^{c}}{\partial x_j}\left(\bar{f} + \bar{g}u_j\right) + \bar{\alpha}_{\rho,\gamma}(h_{ij}^{c}) \geq 0 \\ &2\,L_{\bar{f}}h_{ij}^{c} + L_{\bar{g}}h_{ij}^{c}\,u_i + L_{\bar{g}}h_{ij}^{c}u_j + \bar{\alpha}_{\rho,\gamma}(h_{ij}^{c}) \geq 0 \end{aligned} \tag{4.27}$$

which will be satisfied if both agents i and j satisfy the constraint

$$\dot{h}_{ij}(x_i, x_j) + \frac{\bar{\alpha}_{\rho,\gamma}(h_{ij}(x_i, x_j))}{2} \geq 0. \tag{4.28}$$

In addition, as discussed in Theorem 4, constraint (4.27) will still be satisfied at time

$$T_{ij} \leq \frac{1}{\gamma(1-\rho)}|h_{ij}^{c}(x_{0,i}, x_{0,j})|^{1-\rho}. \tag{4.29}$$

The same argument can be repeated for all pairs $(i,j) \in E_k$, and condition $\mathcal{G}_k \subseteq \mathcal{G}(t)$ will be satisfied within time

$$T_k = \max_{\substack{(i,j)\in E_k \\ \text{s.t. } h_{ij}^{c}(x_{0,i},x_{0,j})<0}} \{T_{ij}\}. \tag{4.30}$$

■

Applying the same design principle described in Section 4.2, the minimally invasive control action can be computed by each robot as

$$u_i^{*} = \arg\min_{u_i \in U_i} \|\widehat{u}_{k-1,i} - u_i\|^2 \tag{4.31}$$

subject to

$$L_{\bar{f}}\,h_{ij}^{c} + L_{\bar{g}}\,h_{ij}^{c}\,u_i + \frac{\bar{\alpha}_{\rho,\gamma}(h_{ij}^{c})}{2} \geq 0, \quad \forall j \in \mathcal{N}_{k-1}^{i} \cup \mathcal{N}_{k}^{i}. \tag{4.32}$$

Similarly to constraint (4.15), once all edges in E_k are available, constraint (4.32) is substituted with

$$L_{\bar{f}}\, h^c_{ij} + L_{\bar{g}}\, h^c_{ij}\, u_i + \frac{\bar{\alpha}_{\rho,\gamma}(h^c_{ij})}{2} \geq 0, \quad \forall j \in \mathcal{N}^i_k, \tag{4.33}$$

which remains active until $\mathcal{B}_k$ is completed.

We note that, in order for agent i to respect (4.33), the only external information needed is the state of all current neighbors, i.e. x_j for all $j \in \mathcal{N}^i_k$. On the other side, in order to respect (4.32), robots need to have access to the state of the future neighbors. This requirement can be satisfied through a state estimation scheme (e.g. EKF [91]), which in turn requires knowledge of robots' dynamics (known for homogeneous teams) or network localization techniques [92].

Remark 3. *The ability of each robot to have access to an estimate of their future neighbors' state does not eliminate the necessity of establishing neighborhood relationships. In fact, a certain proximity structure between robots might be required by desired controllers' performance that cannot be met through state estimations, or by collaboration tasks that require physical interaction between the robots, e.g. collaborative manipulation [87], sharing of resources [86].*

4.3.2 Additional Constraints

In addition to the proximity constraints discussed above, additional limitations might be imposed on the robots' configuration by the mission and the environment. For illustrative purposes, we consider inter-robots collisions and obstacle avoidance. Following the approach described in [89], we encode each pair-wise separation condition through the following barrier certificate

$$h^a_{ij}(x) = \|x_i - x_j\|^2 - D_a^2 \tag{4.34}$$

and the minimum separation D_a between the robots is satisfied if $h^a_{ij}(x) \geq 0$, for all physical neighbors $j \in \mathcal{N}^i(t)$.

Similarly, avoidance of fixed obstacles can be introduce by considering M ellipsoidal regions of the domain, described by centers $o = [o_1^T, \ldots, o_M^T]^T$. For every agent-obstacle pair (i, m) we define a pairwise barrier function as

$$h^o_{im}(x) = (x_i - o_m)^T P_m (x_i - o_m) - 1 \tag{4.35}$$

$$P_m = \begin{bmatrix} a_m & 0 \\ 0 & b_m \end{bmatrix} \quad a_m, b_m > 0. \tag{4.36}$$

The object avoidance constraints are satisfied if $h^o_{im}(x) \geq 0$, for all $i \in V$ and $m \in \{1, \ldots, M\} = \mathcal{I}_M$.

Collecting all the constraints, we expand the problem formulation in (4.14) to

$$\begin{aligned} u_i^* &= \arg\min_{u_i \in U_i} \|\hat{u}_{k-1,i} - u_i\|^2 \\ L_f\, h^c_{ij} + L_g\, h^c_{ij}\, u_i + \frac{\bar{\alpha}_{\rho,\gamma}(h^c_{ij})}{2} &\geq 0, \qquad \forall j \in \mathcal{N}^i_k \\ L_f\, h^s_{ij} + L_g\, h^s_{ij}\, u_i + \alpha(h^s_{ij}) &\geq 0, \qquad \forall j \in \mathcal{N}^i(t) \\ L_f\, h^o_{im} + L_g\, h^o_{im}\, u_i + \alpha(h^s_{ij}) &\geq 0, \qquad \forall m \in \mathcal{I}_M \end{aligned} \tag{4.37}$$

where α is a locally Lipschitz extended class-$\mathcal{K}$ function and the first constraint is replaced by (4.32) during transitions. In conclusion, if there exists a set of control inputs $u = [u_1, \ldots, u_N]$ that simultaneously satisfies all constraints in (4.37), for all behaviors $k = 1, \ldots, M$, Problem 3 will be solved by the robots.

Decentralized Behaviors Sequencing

For the correct execution of the behaviors sequencing, each robot should start assembling a new graph only after all other robots have completed the current behavior. Similarly, a new behavior should start once all robots satisfy the neighbors' requirements for it. Now,

we describe a decentralized strategy that allows execution of these two transitions without the need of a supervisor, nor synchronization between the robots.

With reference to Fig. 4.2, at any given time, each robot's mode of operation is described by a binary variable α_i that describes whether robot i is assembling the graph for an upcoming behavior ($\alpha_i = 1$) or executing a behavior ($\alpha_i = 0$). Without loss of generality, assume robots' initial configuration satisfies the communication requirements for the first behavior, which is then executed ($\alpha_i = 0$). Once all robots have completed the first behavior, they start assembling the graph required by the following one ($\alpha_i = 1$), while minimally perturbing the behavior just concluded. Once the new graph is satisfied $\mathcal{G}_2 \subseteq \mathcal{G}(t)$, robots start behavior $\mathcal{B}_2$ and exit from assembly mode ($\alpha_i = 0$). This process repeats, until no successive behavior exists.

A correct execution of this process requires robot to agree on when to perform transitions $\alpha_i = 0 \to 1$ and $\alpha_i = 1 \to 0$. To this end, we take inspiration from the consensus-based algorithm described in [93] and we note that this choice is not central to the contribution of this chapter. For each robot, we define a binary variable available only to robot i, $s_{t,i} \in \{0, 1\}$ that denotes whether robot i itself has completed its current task $s_{t,i} = 1$ ($s_{t,i} = 0$ if robot has not completed its current task). In addition, we introduce a variable $\sigma_i \in \mathbb{R}_+$, shared among neighbors, continuously updated through the following consensus-based process

$$\sigma_i^+ = s_{t,i} \frac{1}{|\mathcal{N}_i(t)| + 1} \left(\sum_{j \in \mathcal{N}_i(t)} \sigma_j + 1 \right), \tag{4.38}$$

where σ_i^+ represent the variable's value after the update. Owing to the diffusion of $\sigma_1, \ldots, \sigma_N$ throughout the network, we can interpret σ_i's as local measures of the team-wise completion of a task. As proved in [93], if $s_{t,i} = 1$ for all $i = 1, \ldots N$ (i.e., all robots are capable to complete the current behavior), by following (4.38), $\lim_{t \to \infty} \sigma_i = 1$, for all $i = 1, \ldots N$. Therefore, robot i starts assembling a new communication graph once the value of σ_i is close enough to 1 (see [93] for a discussion on how to choose the switching threshold). A similar process is used for the transition $\alpha_i = 1 \to 0$, where we replace $s_{t,i}$ and σ_i with $s_{a,i}$

and η_i respectively. The distributed sequencing procedure is summarized in Algorithm 5.

Algorithm 5 Distributed Composition of behaviors

$\pi \leftarrow \{\mathcal{B}_1, \ldots, \mathcal{B}_M\}$
$k = 1$
$\alpha_i \leftarrow 0$
while $k < M + 1$ **do**
  for $j \in \mathcal{N}_i(t)$ **do**
    $\{X_i, \Sigma_i, H_i\} \leftarrow \{X_i, \Sigma_i, H_i\} \bigcup \{x_j, \sigma_j, \eta_j\}$
  end for
  $\widehat{u}_i \leftarrow \mathcal{U}_k(x_i, X_i)$
  if $\alpha_i = 0$ **then**
    if *Task Complete* **then**
      $s_e \leftarrow 1$
    else
      $s_e \leftarrow 0$
    end if
    $\sigma_i := s_e \frac{1}{|\mathcal{N}_k^i|+1}(\sum_{j \in \mathcal{N}_i(t)} \sigma_j + 1)$
    if $\sigma_i > \bar{\sigma}$ **then**
      $\alpha_i \leftarrow 1$
      $k \leftarrow k + 1$
    end if
  else
    if *Task Complete* **then**
      $s_a \leftarrow 1$
    else
      $s_a \leftarrow 0$
    end if
    $\eta := s_a \frac{1}{|\mathcal{N}_k^i|+1}(\sum_{j \in \mathcal{N}_i(t)} \eta_j + 1)$
    if $\eta > \bar{\eta}$ **then**
      $\alpha_i \leftarrow 0$
      $s_e \leftarrow 0$
    end if
  end if
end while

4.3.3 Applications

We implemented the distributed sequencing framework on the Robotarium [67], on a team of 5 differential drive robots. For this example, controllers are designed assuming a single integrator model, i.e. $\bar{f}(x_i) = [0, 0]^T$ and $\bar{g}(x_i) = I_2$. In this example, robots execute a tran-

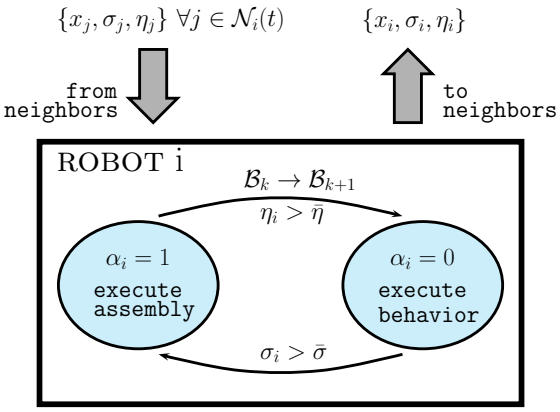


Figure 4.2: Representation of the distributed sequencing framework and information flow. At all times, each robot's state is in either `behavior execution` ($\alpha_i = 0$) or `graph assembly` ($\alpha_i = 0$) modes. Switching between the two modes is triggered by the variables σ_i and η_i whose values is continuously) updated through (4.38). When a switching between `graph assembly` and `behavior execution` occurs, a new behavior is started.

sition between two behaviors, where $\mathcal{B}_1$ is a cyclic-pursuit behavior and $\mathcal{B}_2$ is a formation assembly with leader. Cyclic-pursuit behavior is obtained through the following controller:

$$\hat{u}_i = \sum_{j \in \mathcal{N}_1^i} R(\phi)\,(x_j - x_i) \quad \forall\, i = 1, \ldots, 5,$$

where $R(\phi) \in SO(2)$ is the rotation matrix of angle ϕ, which is related to the desired cycle radius. Importantly, for this behavior to work, the communication graph $\mathcal{G}_1$ must be a cycle graph. Considering robot 1 as leader, the formation control behavior can be achieved with

$$\hat{u}_1 = \sum_{j \in \mathcal{N}_2^i} \left((\|x_i - x_j\|^2 - \theta_{ij}^2)(x_i - x_i) \right) + \gamma_g (x_g - x_i)$$

$$\hat{u}_i = \sum_{j \in \mathcal{N}_2^i} (\|x_i - x_j\|^2 - \theta_{ij}^2)(x_j - x_i) \quad i = 2, \ldots 5$$

where $\theta_{ij} \in \mathbb{R}_+$ is the desired inter-robot distance, $x_g \in \mathcal{D}$ is the leader's goal, and $\gamma_g \in \mathbb{R}_+$ the corresponding proportional gain. In the case of formation control, it is known that the Euclidean embedding of $\mathcal{G}_2$ must be a rigid framework (see for instance [83] and references therein). With reference to Fig. 4.3, robots initially execute $\mathcal{B}_1$ for a certain amount of time

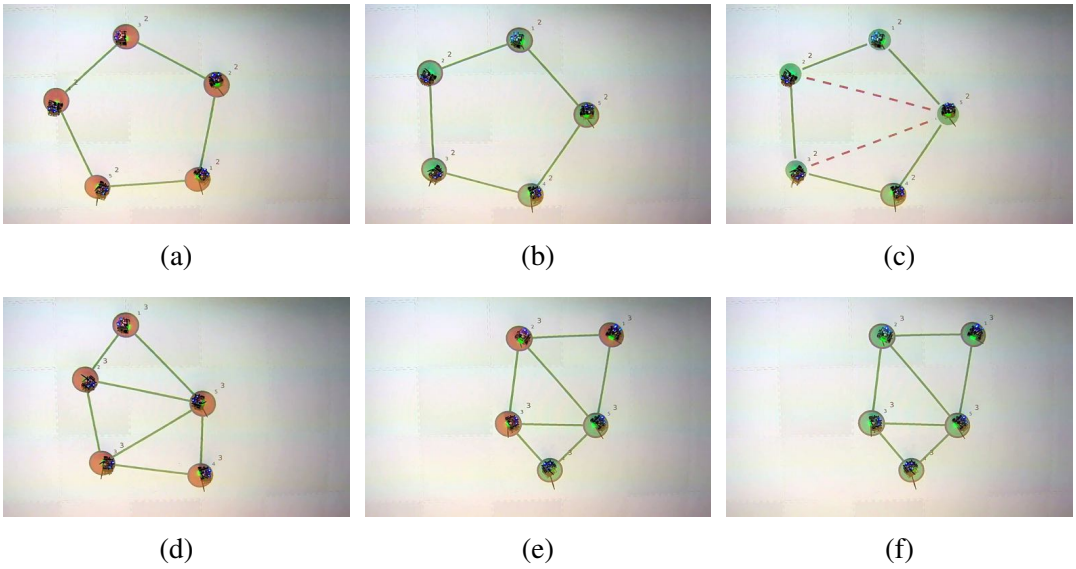


Figure 4.3: Overhead screen-shots from experiments on the Robotarium. Five robots execute two behaviors in sequence (cyclic-pursuit and formation). In figure, green patches represent robots that have completed their task, black rings represent robots that have all neighbors needed for the following task, and green lines represent edges that are available in the current communication graph. From (a) to (b) robots complete the first behavior. During second behavior, additional edges $(2, 5)$ and $(3, 5)$ are required (red dashed line represent missing edges). From (c) to (d), robots $2, 3, 5$ reduce their distance below the communication threshold. After the new graph is complete (d), robots initiate the second behavior (e) and complete it (f).

(a). Once completed (b) (green patches represent robots that have completed their current behavior), robots start assembling $\mathcal{G}_2$ (c), after which $\mathcal{B}_2$ is executed until $\|\hat{u}_i\|$ is below a pre-defined threshold (d-f).

In Fig. 4.4 we can observe the value of the two consensus variables σ_i and η_i for all robots during the behavior transition. Background colors represent the time intervals during which the two behaviors were executed, while the darker region in the middle corresponds to the assembly of the new graph. We observe the assembly and task variables η_i and σ_i approaching the value 1 simultaneously for all robots, thus triggering a synchronized start of the successive phase.

The robustness of our technique was tested by simulating uniformly distributed delays between the robots. Results for this case are shown in Fig. 4.5 where we observe that although convergence of η_i and σ_i is no longer monotonic, robots still reach agreement on

when to switch to the successive phase.

Finally, in order to show the benefits of the minimally invasive approach, we compare it with an alternative technique inspired by [94], where, upon collective completion of a behavior, robots execute rendezvous until the communication graph required by the successive behavior is assembled. As shown by the simulation results for a sequence of 7 behaviors (Fig. 4.6), the mean of the input's norm when considering our framework (red solid line) is always lower than the one obtained using the rendezvous as *glue* behavior. Importantly, since transitions between behaviors occur faster in the minimally invasive case, the lower control effort cannot be attributed to a more *relaxed* choice of controller gains.

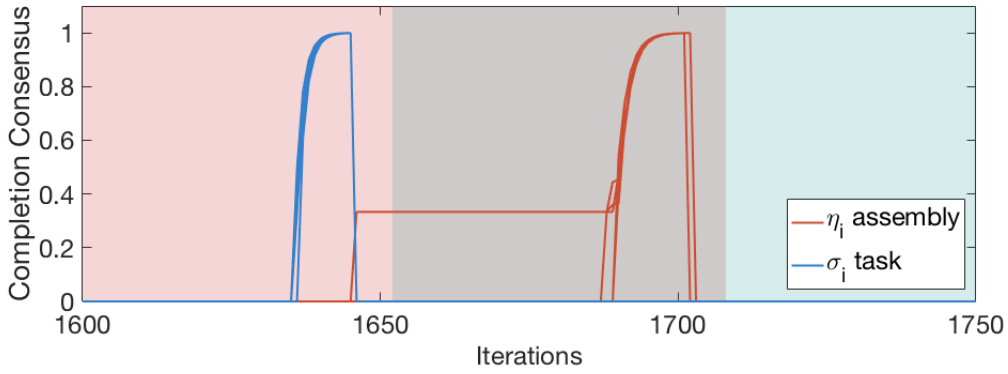


Figure 4.4: Task and assembly consensus variables σ_i and η_i for $i = 1, \ldots, N$ during a transition between two behaviors.

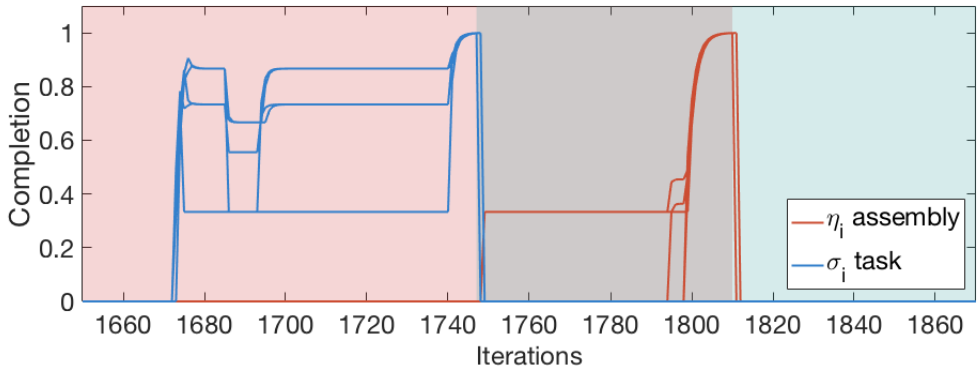


Figure 4.5: Task and assembly consensus variables σ_i and η_i for $i = 1, \ldots, N$ during a transition between two behaviors with communication delays.

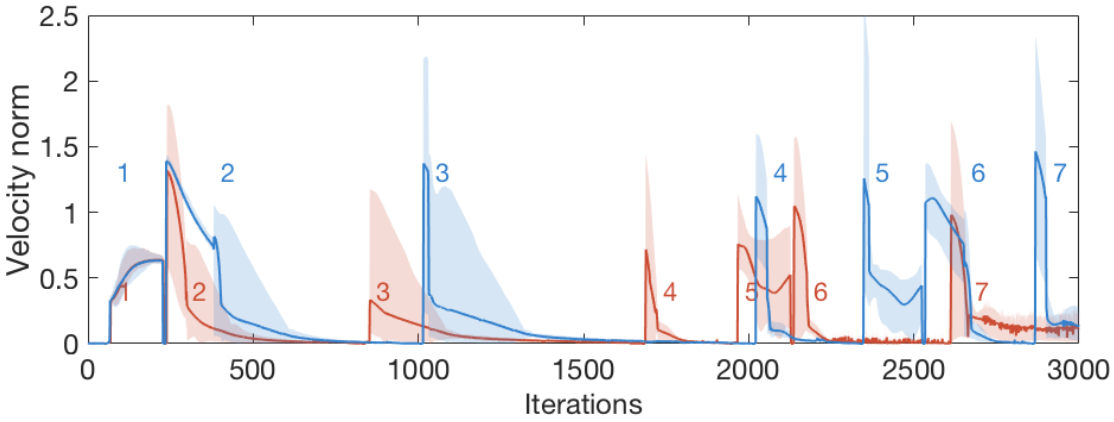


Figure 4.6: Control input comparison between the minimally invasive sequencing framework proposed in this chapter (red) and a sequencing based on rendezvous as *gluing* behavior (blue). Solid lines represent the mean of the control input across all robots, while shaded regions represent the interval between minimum and maximum control input.

4.4 Case Study: Securing a Building

The objective of this section is to describe the *Securing a Building* mission, which will be used as testing scenario for the composition framework. We describe now the main structure and objective of the mission, while we deconstruct it into coordinated behaviors in the next subsection.

4.4.1 Mission Overview

In the Securing a Building mission, a group of robots are deployed in an urban environment to identify an unknown target building and rescue a subject located inside. Based on [95], we decompose this mission into the following 4 phases:

FIND - First, the robots are tasked with identifying the target building by means of surveillance of the perimeters of all the buildings. For efficient exploration, robots can be broken into sub-teams. Each team reports collected information at the base after each building has been investigated. Once the target building has been identified, the robots reunite and prepare for the next phase.

ISOLATE - The robots isolate the target building by patrolling its perimeter. To achieve

this, the robots are divided into two subgroups - the *security agents* responsible for boundary protection and the *maneuvering agents* tasked with entering the building.

RESCUE - During the rescue phase, the security agents keep patrolling around the building. In the meanwhile, the maneuvering agents enter the building, clear the rooms, and seize positions as they maneuver through the building to find the subject to be rescued. Once the subject has been located, the robots transport it to the safe zone.

FOLLOW-THROUGH - As the interior of the building is being cleared, individual robots are left inside as beacons, while the remaining robots from the maneuvering agents leave the building, gather on the outside with the security agents, and report back to the base station.

A number of arguments support the choice of the Securing a Building mission as an ideal scenario for testing multi-robot techniques and algorithms. First, the requirement of spatially diverse functionalities that cannot be provided by single robots naturally requires the use of multi-robot systems. Second, the final goal of the mission, namely rescuing the subjects of interest, reflect the fact that general real-world missions cannot be accomplished with single controllers. Lastly, thanks to its modularity, techniques focusing on specific aspects of the mission can be integrated and tested without influencing the overall structure of the mission (see the Appendix for details).

4.4.2 Securing a Building Through Composition of Behaviors

We deconstruct the Securing a Building mission through ordered sequences of coordinated behaviors. The process is summarized in Fig. 4.7. We refer to behaviors in terms of their main objectives, acknowledging that different implementations can be used to achieve the same results. We highlight these behaviors in parenthesis.

FIND Robots initially coordinate with the operator at the base station (*rendezvous*). After that, robots are divided into different search teams, each assigned with a list of buildings

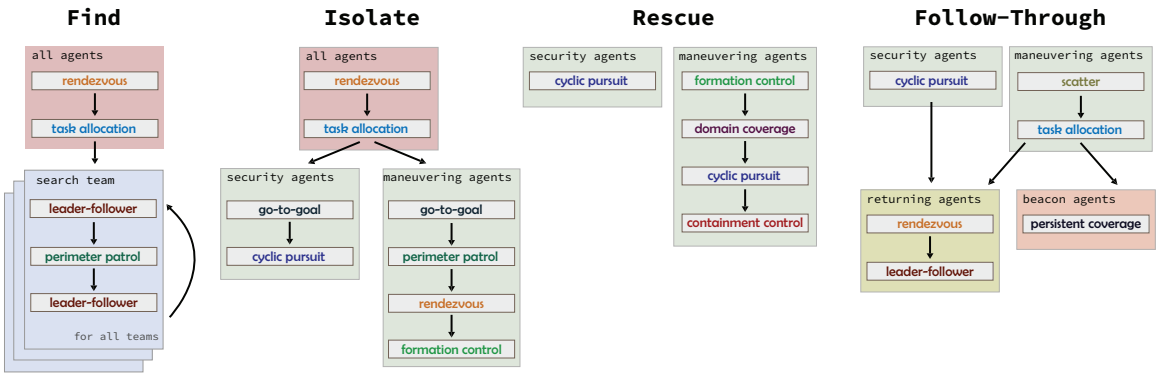


Figure 4.7: Mission design chart showing how coordinated behaviors are composed together to tackle the Securing a Building mission. The four bold titles are the mission phases and the large boxes below them indicate specific agent roles and associated behaviors. The arrows in the chart indicate the transitions between different behaviors. We note that he choice of controllers that produces the behaviors in the chart is not unique.

to investigate (*task allocation*). Subsequently, all the teams investigate their own lists of buildings. First team of robots travels to the vicinity of a building (*leader-follower*), then start to survey the exterior of the building (*perimeter patrol*), and return to the base (*leader-follower*). This process repeats until the target building is discovered.

ISOLATE Robots gather near the base (*rendezvous*), then are divided into *security* and *maneuvering* agents (*task allocation*). After traveling from the base to the vicinity of the target building (*go-to-goal*), security agents protect the building's perimeter (*cyclic pursuit*), until the end of the RESCUE phase. Meanwhile, the maneuvering agents locate the building's entrance, by following its perimeter (*perimeter patrol*). Once the entrance has been found, the maneuvering agents gather at the entrance (*rendezvous*) and create a formation (*formation control*) before entering.

RESCUE The maneuvering agents enter the building in formation (*formation control*) and cover the interior area (*area coverage*). Once the location of the subject to rescue is identified, the robots form a circular closure around the subject (*cyclic pursuit*). Then, the robots transport the subject to the safety zone, while maintaining the circular closure around the subject (*containment control*).

FOLLOW-THROUGH Maneuvering agents spread (*scatter*) over the interior of the building. To signify that the area has been cleared, few robots are left inside the building as beacons (*persistent coverage*). The rest of the maneuvering agents and the security agents reunite outside the building (*rendezvous*). At last, they return to the base (*leader-follower*).

4.4.3 Results

We tested the behavior composition framework described in Section 4.3 on the Securing a Building mission, which was executed on the Robotarium [67]. In Fig. 4.8, we display selected snapshots of the mission obtained by a camera mounted on the ceiling. In the experiment, 8 differential-drive robots, indexed $1, \ldots, 8$ are deployed in a simulated urban environment composed of 6 buildings, blue polygons indexed $1, \ldots, 6$. In this experiment, we simulate a maximum sensor range $\Delta = 0.5$m. Because of the different spatial scales between FIND/ISOLATE phases and RESCUE/FOLLOW-THROUGH phases, the entire mission is divided in two parts. In the first part (Fig. 8.2a to Fig. 4.8d) the experiment is performed at a *neighborhood*-level scale. The remaining two phases are executed in a zoomed-in environment, which focuses on the one building of interest (Fig. 4.8d to Fig. 4.8f).

During FIND phase (Fig. 8.2a and 8.2b), two groups of robots $\text{TEAM}1 : \{1, 2, 3, 4\}$ and $\text{TEAM}2 : \{5, 6, 7, 8\}$ investigates preassigned lists of buildings, leaving some agents near the base station (the purple filled dot in the top right corner) if destination building cannot be reached without breaking the connectivity constraints. The red polygon in Fig. 8.2b and 4.8c is the target building after being identified by $\text{TEAM}1$. During the ISOLATE phase (Fig. 4.8c), maneuvering agents look for the entrance, while the security agents secure the outer perimeter.

During the RESCUE phase (Fig. 4.8d to 4.8e), the agents inside the building, i.e. $\text{TEAM}1$, localize the target (red dot) using Voronoi coverage (Fig. 4.8d) and escort it to the safe area (red circle) as shown in (Fig. 4.8e). Finally, robots 1 and 2 are left as beacon

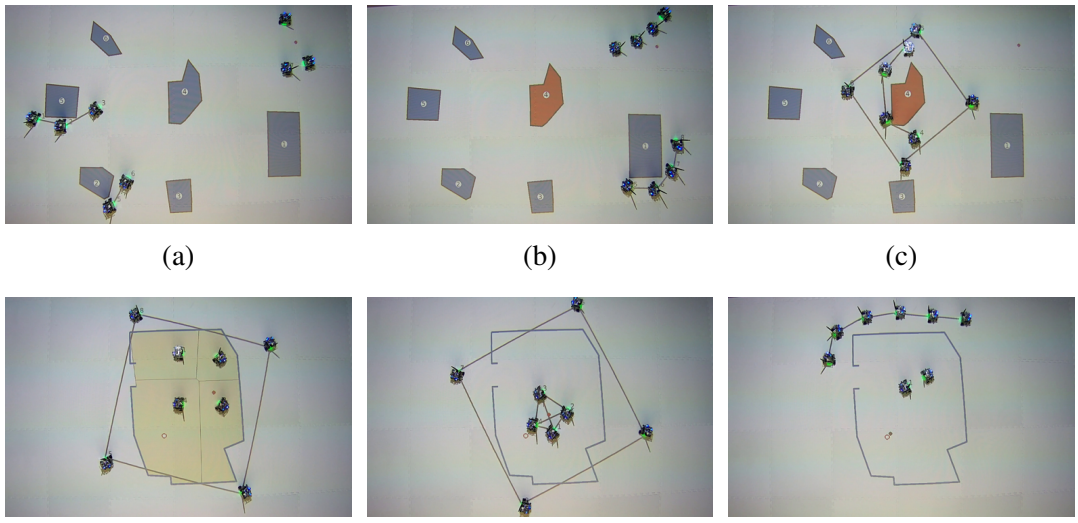


Figure 4.8: Overhead screen-shots from experiments on the Robotarium. A team of eight robots is divided in TEAM1 : $\{1, 2, 3, 4\}$ and TEAM2 : $\{5, 6, 7, 8\}$. Because of the different spatial scales between FIND/ISOLATE phases and RESCUE/FOLLOW-THROUGH phases the mission is executed on two different environments. Each team is assigned with a list of three buildings to inspect sequentially. FIND: (a) perimeter patrol of buildings 2 and 5; (b) building 4 is identified as the target building, while TEAM1 waits for TEAM2 to return to base. ISOLATE: (c) TEAM2 secures perimeter of building, while TEAM1 inspects exterior of building, searching for the entrance. RESCUE: after entering the building, TEAM1 performs domain coverage of the building until target (red dot) is identified (d); after this, (e) robots escort target to safe location (red circle). FOLLOW-THROUGH: finally, two robots are left as beacons inside the building while all remaining robots return to base (f).

inside the building, while remaining robots return to the base (Fig. 4.8f).

4.5 Key Assumptions in Barrier Function-based Frameworks

When using quadratic programs of the form (1.6) in order to execute complex tasks such as the one described in this chapter, we need to account for failure modes and scenarios such as infeasibility type situations which can lead to a dissatisfaction of the task specification. Hence, the following are some key observations and assumptions that are inherently embedded in standard barrier function-based quadratic programming approaches. Such assumptions need to be addressed when executing complex tasks such as the one described in this chapter.

- **Feasibility of QP:** This is a prevalent assumption in many existing barrier function based works, and is an important point that must be addressed when one uses these frameworks for safety-critical tasks. The next chapter (Chapter 5 and Chapter 6) will describe some techniques to relax the barrier function constraints while guaranteeing feasibility.

- **Negligence of System Volume:** Traditional barrier function formulations guarantee safety of the system state. However, in reality, systems have a volume associated with them. Hence, providing formal guarantees on safety of the system volume is important. We discuss a special class of control barrier functions introduced in Chapter 7 that guarantee safety of the system volume.

- **Known Environments:** In the above detailed example, we assumed knowledge of the location and shape of the obstacle i.e. complete knowledge of the barrier function (1). However, in many scenarios, this need not necessarily be true. We discuss such an instance in Chapter 8, and provide a general framework to synthesize control barrier functions using sensory input from the environment and techniques from machine learning

Part II

Feasibility Techniques

CHAPTER 5

RELAXED BARRIER CERTIFICATE FORMULATIONS

In the previous two chapters, we described frameworks where each reachability and/or invariance constraint was encoded separately in quadratic programs using finite time and/or zeroing control barrier functions. In addition, the quadratic program was strict; that is, no relaxation of the constraints was introduced. Such optimization based approaches might turn out to be too strict in certain scenarios, and hence, in this chapter we discuss techniques to relax the constraints and the quadratic programs.

Given a task specification, in this section we focus on scenarios where using existing methods in literature [8], [9], [96] will render the controller infeasible, and provide solutions for the same. The first section discusses Theorem 6 which appeared in our conference paper [32], while the second section proposes a relaxed formulation of the QP based controller in scenarios where the invariance constraints encoded by zeroing barrier functions are in conflict.

5.1 Composite Finite Time Control Barrier Functions

Consider two robots R_1 and R_2 as shown in the workspace in Fig 5.1. Suppose R_1 is sensing information from R_2 and hence must always stay within the sensing radius of R_2. Suppose we have two regions of interest A, and B. Let $\mathcal{D}$ represent a corridor in the state space (denoted by the dotted lines in Fig 5.1) where R_1 must maintain a very small distance of connectivity with R_2. This could represent, for example, an area with very poor network connectivity and hence the robots must resort to communication over small distances. Let the specification for the multi-agent system be given as $\phi = \Diamond(\pi_1^A \wedge \pi_2^B) \wedge \Box\pi_{conn}$ where π_1^A is the proposition that is true when R_1 is in A, π_2^B is the proposition that is true when R_2 is in B, and π_{conn} is the proposition that is true when the robots must maintain connectivity

at all times. In other words, R_1 must visit A, R_2 must visit B and R_1 must always stay connected with R_2. The workspace is as shown in Fig 5.1.

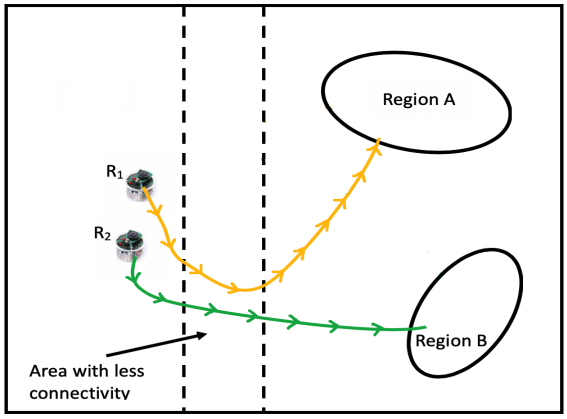


Figure 5.1: Representative trajectories for R_1 and R_2 that satisfy the specification $\phi = \Diamond(\pi_1^A \wedge \pi_2^B) \wedge \Box\pi_{conn}$. The area with less connectivity is the corridor $\mathcal{D}$. Observe that R_1 and R_2 need to maintain a small distance of connectivity within the corridor D.

By following the method proposed in [9], the QP that is to be solved is as follows:

$$\begin{aligned} \min_{u\in\mathbb{R}^4} \quad & \|u\|_2^2 \\ \text{s.t} \quad & L_f h_A(x_1) + L_g h_A(x_1)u \geq -\gamma \cdot \text{sign}(h_A(x_1)) \cdot |h_A(x_1)|^\rho \\ & L_f h_B(x_2) + L_g h_B(x_2)u \geq -\gamma \cdot \text{sign}(h_B(x_2)) \cdot |h_B(x_2)|^\rho \\ & L_f h_{conn}(x) + L_g h_{conn}(x)u \geq -\alpha(h_{conn}(x)) \end{aligned} \tag{5.1}$$

where α is a locally Lipschitz extended class κ function, $\gamma > 0$, $\rho \in [0, 1)$, x_1 is the state of R_1, x_2 is the state of R_2, $x = [x_1\ x_2]$ is the total state of the system, h_A is the FCBF which represents A, h_B is the FCBF which represents B, and h_{conn} is the ZCBF which dictates the connectivity radius to be maintained by R_1 with R_2. Note that [9] considers only reachability tasks and not more general LTL tasks. In addition, [9] requires multiple reachability specifications to be encoded as separate constraints in a QP, as discussed above.

However, this QP becomes infeasible at the point when R_1 and R_2 reach the corridor $\mathcal{D}$. This is because the first constraint in (5.1) dictates that R_1 make progress towards A, but the third constraint dictates that R_1 move closer to R_2 and hence move away from A.

This leads to an empty solution space thus rendering the QP infeasible. This shows that the above formulation of encoding multiple reachability objectives as individual constraints is too restrictive.

In light of the above scenario, we propose a method in which we compose multiple FCBFs. By ensuring that the total sum of the finite time barrier functions is always increasing, we can allow for decrease in the values of some of the individual barrier functions thereby allowing some robots to move away from their desired sets temporarily. This provides a larger solution space for the QP. This is formalized in the following theorem.

Theorem 6. *Consider a dynamical system in control-affine form as in* (1.1)*. Given* $\Gamma \subset \mathcal{X}$ *defined by a collection of* $q \geq 1$ *functions* $\{h_i(x)\}_{i=1}^{q}$ *such that* $\Gamma = \bigcap_{i=1}^{q} \{x \in \mathcal{X} \mid h_i(x) \geq 0\}$ *and for* $i = \{1, 2, 3, ..., q'\}$ *with* $q' < q$*,* $h_i(x)$ *is bounded i.e.* $h_i(x) < M_i$ *for all* $x \in \mathcal{X}$*, for* $M_i > 0$.[1] *If there exists a collection* $\{\alpha_i\}_{i=1}^{q'}$ *with* $\alpha_i \in \mathbb{R}_{>0}$*, parameters* $\gamma > 0$*,* $\rho \in [0, 1)$ *and a continuous controller* $u(x)$ *where* $u : \mathcal{X} \to \mathbb{R}^m$*, such that for all* $x \in \mathcal{X}$

$$\sum_{i=1}^{q'} \Big\{ \alpha_i (L_f h_i(x) + L_g h_i(x) u(x)) \Big\} + \gamma \cdot \textit{sign}\Big(\min \Big\{ h_1(x), h_2(x), \ldots, h_{q'}(x) \Big\} \Big) \geq 0 \quad (5.2)$$

$$L_f h_i(x) + L_g h_i(x) u(x) + \gamma \cdot \textit{sign}(h_i(x)) \cdot |h_i(x)|^{\rho} \geq 0 \,, \; \forall \; i \in \{q' + 1, \ldots, q\} \quad (5.3)$$

then under the feedback controller $u(x)$*, for all initial conditions* $x_0 \in \mathcal{D}$*, there exists* $0 < T < \infty$ *such that* $x(T) \in \Gamma$.

Proof. By contradiction, suppose for some $x_0 \in \mathcal{X} \backslash \Gamma$ the control law $u(x)$ that satisfies (5.2) and (5.3) is such that there does not exist a finite time $0 < T < \infty$ so that $x(T) \in \Gamma$. In particular, then for all $t > 0$, $min\Big\{ h_1(x(t)), h_2(x(t)), \ldots, h_q(x(t)) \Big\} < 0$, where $x(t)$ is the solution to (1.1) initialized at $x(0)$ under the control law $u(x)$. By

[1] If all the functions are bounded, then $q' = q$ and so we will have only (5.2) as a constraint in the QP $\forall i \in \{1, 2, \ldots, q\}$

(5.3) for all $t > T_i = \frac{|h_i(x_0)|^{1-\rho}}{\gamma(1-\rho)}$, we have $h_i(x(t)) \geq 0$ for all $i = \{q'+1, \ldots, q\}$ by Proposition 1. To that end, if we define $T' = \max_{i=q'+1,\ldots,q} \{T_i\}$, then for all $t > T'$ we have, $\min\left\{h_1(x(t)), h_2(x(t)), \ldots, h_{q'}(x(t))\right\} < 0$. In particular, observe that

$$\frac{d}{dt}\sum_{i=1}^{q'}\left\{\alpha_i h_i(x(t))\right\} = \sum_{i=1}^{q'}\left\{\alpha_i(L_f h_i(x) + L_g h_i(x)u(x))\right\} \tag{5.4}$$

so that by integration of (5.4) using the fundamental theorem of calculus and (5.2), we have

$$\sum_{i=1}^{q'}\left\{\alpha_i h_i(x(t))\right\} \geq \gamma \cdot (t - T') + \sum_{i=1}^{q'}\left\{\alpha_i h_i(x(T'))\right\}.$$

We observe that as $t \to \infty$, $\sum_{i=1}^{q'}\left\{\alpha_i h_i(x(t))\right\} \to \infty$. But this is a contradiction since $h_i(x(t))$ for $i = \{1, 2 \ldots, q'\}$ is bounded i.e. $\sum_{i=1}^{q'}\left\{\alpha_i h_i(x(t))\right\} < \sum_{i=1}^{q'} \alpha_i M_i$. Thus, there exists a $0 < T < \infty$ such that $x(T) \in \bigcap_{i=1}^{q} \{x \in \mathcal{X} \mid h_i(x) \geq 0\}$. ■

Theorem 6 allows a system to reach an intersection of multiple regions in the state space using a single barrier certificate constraint. In contrast, [9] proposes a more restrictive solution to the constrained reachability problem with desired level sets being individually defined by multiple FCBFs in a QP. In particular, [9] proposes the set of control laws $\underline{\mathcal{U}}$ given by

$$\underline{\mathcal{U}}(x) = \{u \in \mathbb{R}^m \mid L_f h_i(x) + L_g h_i(x)u + \gamma \cdot \text{sign}(h_i(x)) \geq 0\ ,\ \forall i \in \{1, \ldots, q\}\}. \tag{5.5}$$

Note that this is equivalent to taking $q' = 0$ in Theorem 6. Define

$$\mathcal{U}(x) = \{u \in \mathbb{R}^m \mid \text{(5.2) and (5.3) are satisfied}\}, \tag{5.6}$$

and $\widehat{\Gamma} = \bigcup_{i=1}^{q} \{x \in \mathcal{X} \mid h_i(x) \geq 0\}$. We then formulate the following corollary.

Corollary 1. *Under the hypotheses of Theorem 6 with* $\sum_{i=1}^{q'} \alpha_i \geq 1$*, the set* $\mathcal{U}(x)$ *defined in* (5.6) *is a superset to the set* $\underline{\mathcal{U}}(x)$ *defined in* (5.5)*. That is,* $\underline{\mathcal{U}}(x) \subseteq \mathcal{U}(x)$ *for all* $x \in \mathcal{X}\backslash\widehat{\Gamma}$.

Proof. Note that for $q' = 0$ and $q' = 1$, it follows that $\underline{\mathcal{U}}(x) = \mathcal{U}(x)$. Now, for any $q' \in \{2, \ldots, q-1\}$, consider any $u(x) \in \underline{\mathcal{U}}(x)$ applied to the system (1.1). Hence, we have that

$$L_f h_i(x) + L_g h_i(x) u(x) \geq -\gamma \cdot \text{sign}(h_i(x(t))) \geq \gamma$$
$$\alpha_i (L_f h_i(x) + L_g h_i(x) u(x)) \geq \alpha_i \cdot \gamma$$

for all $i \in \{1, 2, \ldots, q\}$ since $x \in \mathcal{X}\backslash\widehat{\Gamma}$. Summing the inequalities for the barrier functions from $i = 1, 2, \ldots, q'$, we get

$$\sum_{i=1}^{q'} \left\{ \alpha_i (L_f h_i(x) + L_g h_i(x) u(x)) \right\} \geq \sum_{i=1}^{q'} \alpha_i \cdot \gamma \geq \gamma$$

where the last inequality follows by assumption that $\sum_{i=1}^{q'} \alpha_i \geq 1$. This implies $u(x)$ satisfies (5.2) since $\text{sign}\left(\min \left\{ h_1(x), h_2(x), \ldots, h_{q'}(x) \right\} \right) < 0$ for all $x \in \mathcal{X}\backslash\widehat{\Gamma}$, and $L_f h_i(x) + L_g h_i(x) u(x) \geq -\gamma \cdot \text{sign}(h_i(x(t)))$ for all $i \in \{q'+1, \ldots, q\}$ which implies $u(x)$ also satisfies (5.3). Hence, $u(x) \in \mathcal{U}(x)$. Thus the corollary follows. ∎

The choice of α_i is user dependent, and in general we choose $\alpha_i = 1$ for all $i = 1, 2, \ldots, q'$. Corollary 1 is meant to characterize the condition at which Theorem 6 results in a larger feasible solution space than traditional methods. Theorem 6 allows for additional directions of evolution for the system state thereby resulting in a more relaxed approach to the finite time reachability problem.

In particular, the significance of this result can be seen from the plots in Fig 5.2 obtained from the experimental case study in Chapter 3. In Chapter 3 Section 3.3, we used Theorem 6 in order to guarantee feasibility. Fig 5.2 illustrates the significance of Theorem 6. Observe that since we enforce that the sum of the barrier functions must increase,

robot 1 is able to move away from target 1, without leading to an infeasible solution. Note that infeasibility occurs if one were to encode the constraints as per existing literature since that requires that robot 1 keep moving towards target 1 at all time instants. In the code repository[2], we provide a simple example to illustrate such an infeasibility scenario which reinforces the significance of Theorem 6.

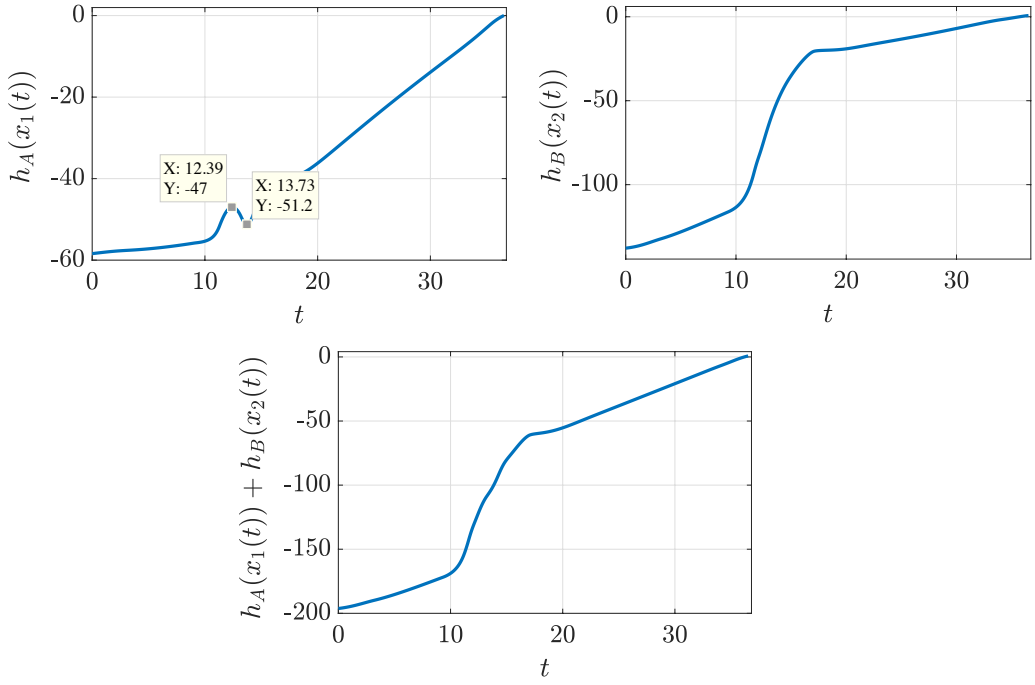


Figure 5.2: The figures above show the progress of robot 1 towards goal region A, the progress of robot 2 towards goal region B, and the total sum of the progress of the robots. Observe that the sum total of the barrier functions is increasing, which guarantees feasibility, even though robot 1 moves away temporarily from A. This is a result of the application of theorem 6 in the QP

5.2 Prioritization of Zeroing Control Barrier Functions

In this section, we introduce a methodology for prioritizing different ZCBFs. In particular, our proposed formulation is similar to [97] where different tasks represented by multiple ZCBFs are prioritized for a multi-agent system. In particular, we stipulate that the perfor-

[2]Code for Robotarium experiment- `https://bit.ly/37QkOBS`

mance objective dictated by the FCBFs must never be relaxed, and hence, they are encoded as *hard* constraints (i.e. constraints which must never be violated), whereas the ZCBFs can be relaxed and hence, they are encoded as *soft* constraints (i.e. constraints which could potentially be violated). Our proposed method is different from [97] in the sense that, in addition to the ZCBFs, we also incorporate FCBFs which are treated as hard constraints in the QP based controller.

Consider the following motivating example. Suppose we have a goal region $\mathcal{G} = \{x \in \mathcal{X} | h_{\mathcal{G}}(x) \geq 0\}$ where $h_{\mathcal{G}} : \mathcal{X} \to \mathbb{R}$ is a FCBF, encapsulated by an obstacle $\mathcal{O} = \{x \in \mathcal{X} | h_{\mathcal{O}}(x) \leq 0\}$ where $h_{\mathcal{O}} : \mathcal{X} \to \mathbb{R}$ is a ZCBF. Suppose the specification for the robot is $\phi = \Diamond\mathcal{G} \wedge \Box\neg\mathcal{O}$. By following the method proposed in existing works such as [98], [99], [5], [9], the QP that is to be solved is as follows:

$$\begin{aligned} \min_{u \in \mathbb{R}^m} \quad & \|u\|_2^2 \\ \text{s.t} \quad & L_f h_{\mathcal{G}}(x) + L_g h_{\mathcal{G}}(x) u \geq -\gamma \cdot \text{sign}(h_{\mathcal{G}}(x)) \cdot |h_{\mathcal{G}}(x)|^{\rho} \\ & L_f h_{\mathcal{O}}(x) + L_g h_{\mathcal{O}}(x) u \geq -\alpha(h_{\mathcal{O}}(x)) \end{aligned} \tag{5.7}$$

where $\gamma > 0$, $\rho \in [0, 1)$ and α is a locally Lipschitz extended class κ function.

However, since the goal is encapsulated by the obstacle, the two constraints are in conflict and hence the QP will be infeasible. In order to tackle scenarios such as the one above, we propose a relaxed formulation of the QP similar to the one in [8], [97].

Consider p ZCBFs and n FCBFs. Let $\mathcal{P}$ be the index sets for the zeroing barrier functions. Some or all of the ZCBFs may be in conflict with the composite FCBF. The generalized relaxed QP is of the form,

$$\begin{aligned} \min_{v = \left[u^T, \epsilon_1, \ldots, \epsilon_p\right]^T \in \mathbb{R}^{m+p}} \quad & \|u\|_2^2 + \frac{1}{2} \Xi^T W \Xi \\ \text{s.t} \quad & \text{(5.2) holds} \\ & L_f h_i(x) + L_g h_i(x) u(x) \geq -\alpha_i(h_i(x)) - \epsilon_i \ , \ \forall i \in \mathcal{P} \end{aligned} \tag{5.8}$$

where $\Xi = \begin{bmatrix} \epsilon_1, \epsilon_2, \ldots, \epsilon_p \end{bmatrix}^T \in \mathbb{R}^p$, $W \in \mathbb{R}^{p \times p}$ is a diagonal matrix with the diagonal elements as $(w_1, w_2, \ldots, w_p)$ where $w_i \in \mathbb{R}_{>0}$ is a weight associated with the the slack variable ϵ_i for all $i \in \{1, 2, \ldots, p\}$, and α_i is a locally Lipschitz extended class κ function. The weight matrix W allows one to encode the notion of "priority" for the barrier functions. For example, if the weight w_i corresponding to the slack variable ϵ_i is large, then the i^{th} ZCBF has higher priority over other constraints.

Remark 4. *Similar to the discussion in Remark 2 in [8], if the reachability and invariance constraints are not in conflict, then with an appropriate choice of the weight matrix W, we will have $\epsilon_i \approx 0$ for some $i \in \{1, 2, \ldots, q\}$. Also, note that we extend the formulation provided in [8] from two constraints to multiple constraints.*

The relaxed QP returns a control law that allows the system to reach the desired level set in a finite time while minimally violating the invariance constraints if there is a conflict with the FCBF. This type of relaxed controller is useful when trying to find minimally violating control strategies for conflicting LTL specifications [100]. We present the following case study which uses the relaxation based controller (5.8).

Example

Consider an omnidirectional robot with dynamics $\dot{x} = u$ where $x \in \mathcal{X} \subset \mathbb{R}^2$, and $u \in \mathbb{R}^2$. Let $\mathcal{D} \subset \mathcal{X}$ be a compact domain in the state space. The workspace is as shown in Fig 5.3. Suppose we have two unsafe regions A and B and a goal region C. Let C be contained within A and B. Suppose the specification to be satisfied by the robot is $\phi = \Diamond C \wedge \Box(\neg A \wedge \neg B)$. From Fig 5.3, we observe that satisfaction of ϕ is impossible without entering the regions A or B. However, suppose that region A has greater priority than region B and hence violation of B is allowed to some extent.

With this additional flexibility, we can employ the proposed QP as in (5.8) with the weights $w_A \in \mathbb{R}_{>0}$ set to be a large value and $w_B \in \mathbb{R}_{>0}$ set to be a small value. We then

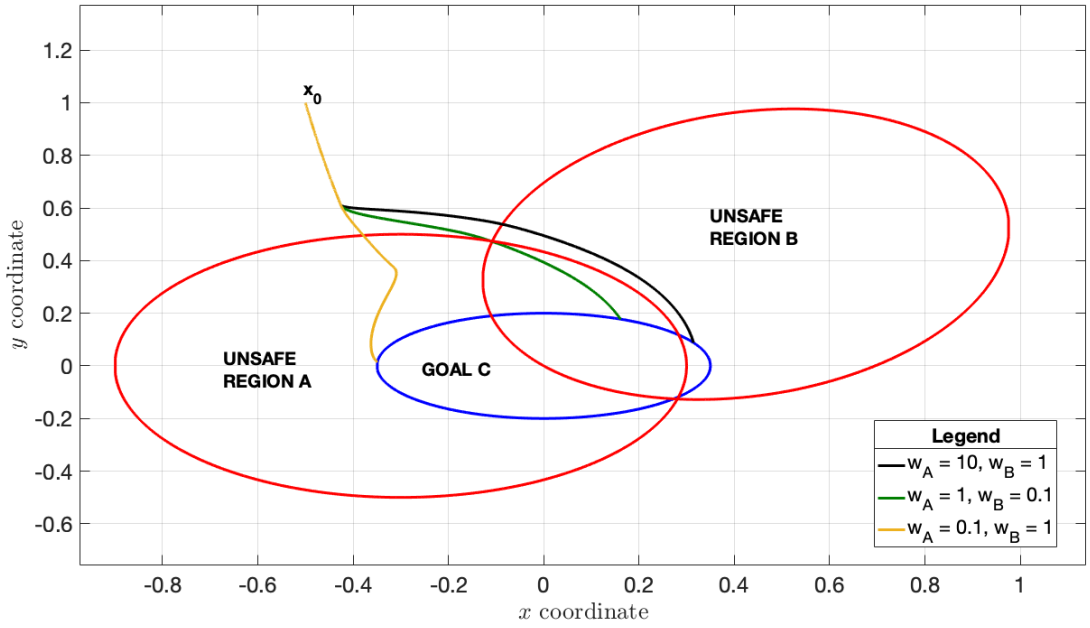


Figure 5.3: A family of trajectories for the robot generated by the relaxed QP (5.8). By changing the values of the entries in the weight matrix W, one can encode the notion of priority for different regions in the state space as can be seen from the various trajectories.

solve (5.8) which gives us a family of trajectories (depending on the values of the weights w_A and w_B) of the robot as shown in Fig 5.3. Observe that with different weights w_A and w_B for the regions in the QP, we obtain a different trajectory. This allows one to also encode the notion of priority in the QP.

5.3 Concluding Remarks

In this chapter, two techniques to relax traditional barrier function formulations were presented. In the first technique, we composed multiple finite time barrier functions into single constraint, which allows for a larger solution space than the traditional method of encoding each constraint separately in a quadratic program. We illustrate the usefulness of the result in reference to the implementation results presented in the earlier chapter. The next technique details a relaxed quadratic program structure, where slack variables are introduced in order to prioritize the zeroing barrier function constraints. We provide simulation results to show a family of trajectories that result from such a weighting of constraints.

CHAPTER 6

WEIGHTED POLAR FINITE TIME CONTROL BARRIER FUNCTIONS

Introduction

In this chapter, we consider the problem of a unicycle robot converging to a region of interest with a desired heading angle within a finite time using control barrier functions. This problem has applications in multi-agent systems. For example, in caging [101], a team of agents must surround an object of interest in order to apply contact forces to the surface of the object and transport it from one point to another. In formation control based surveillance [102], a group of agents must converge to a terminal level sets and then perform patrolling around this desired region. In [103], the authors discuss a game between intruders and defenders where the intruders score points if they reach the target that is being defended by the defenders. This requires the defenders to detect the intruders and perform surveillance effectively. In these applications, it is preferable to have a control policy which guarantees finite time convergence to desired regions with a reasonable bound on the heading angles of the robots.

In the traditional finite time barrier function formulations, the barrier function is only a function of the position of the virtual control point of the robot. This is convenient when the finite time reachability requirement is restricted to the position of the CoM. However, if one requires stricter convergence guarantees such as convergence to a terminal level set with a desired angle, then using the barrier functions suggested in [9] and [32] will not work. In such a situation, it becomes imperative to consider a FCBF defined in the entire state of the system and not just the position. In this chapter, a new finite time control barrier function is introduced for differential drive mobile robots (unicycle dynamics) inspired by the generalization of the weighted L_p norm in [104] in order to incorporate the entire state of the system.

The weighted L_p norm has been recently used for collision avoidance between two full body objects in [104], [105], [5], [106]. In [5], the distance between two quadrotors is defined by measuring the weighted L_p distance of the center of mass of each robot. Here, the orientation of the robot is ignored. In [105], more geometric settings of collision detection are introduced where the distance involves the orientation of both objects and is used to plan a collision avoidance trajectory for a snake robot in [106]. In [104], a weighted L_p method for collision checking is extended to bendable 3D objects (non convex) where the surface is approximated by the one level set of the so-called *weighted polar* L_p *function*. In this chapter, a modification to the weighted polar L_p function is considered which enables us to define a nonconvex terminal level set with orientation constraints. The super zero level set of the proposed barrier function not only captures desired goal positions but also provides a desired angle of arrival for the robot.

We adopt a QP based framework [19] which incorporates the barrier functions as constraints. One of the main difficulties associated with a barrier functions based QP is proving feasibility at all points in the state space. In the case of single integrator dynamics as considered in [98], [99], [9], feasibility is assured due to the simple dynamics. However, when one considers complex nonholonomic systems such as the unicycle dynamics, then proving feasibility is not straightforward. If the framework suggested in [98], [99], [9], [32] is used, then we will encounter points of singularity in the state space where the constraints are invariant to the control. This happens when the Lie derivative of the FCBF along the actuation vector field vanishes. When the system enters such sets, the QP becomes infeasible. In this chapter we identify the condition at which the Lie derivative vanishes. Then, we characterize the appropriate value of the control parameter that must be used in order to eliminate a vanishing Lie derivative condition. This has two important implications. First, this leads to the conclusion that the candidate L_p functions are indeed valid finite time control barrier functions. Second, the QP based controller always returns a feasible solution.

The contributions of this chapter are three fold. First, we introduce a new class of finite time control barrier functions for unicycle robots in the planar space. Our proposed work incorporates the full state of the system as opposed to [98], [9], [32] where only a part of the state is abstracted in the barrier function. This allows us to accomplish stricter convergence specifications such as converging to the desired level set with a desired heading angle. Second, since these barrier functions cannot be used directly with the dynamics in a QP framework due to the existence of the singular sets where the QP is infeasible, we leverage the structure of the proposed FCBF which guarantees feasibility of the QP. Last, we detail a control architecture that allows for finite time convergence to a level set at the desired terminal angle.

6.1 Problem Setup

In this chapter, we consider a differential drive mobile robot with the unicycle dynamics

$$\begin{aligned} \dot{x} &= v \cdot \cos(\phi) \\ \dot{y} &= v \cdot \sin(\phi) \\ \dot{\phi} &= \omega \end{aligned} \tag{6.1}$$

where $x, y \in \mathbb{R}$ are the x and y coordinates of the robot, $\phi \in (-\pi, \pi]$ is the heading, $v \in \mathbb{R}$ is the linear velocity, and $\omega \in \mathbb{R}$ is the angular velocity. Denote the domain of the state space as $\mathcal{X} \subset \mathbb{R}^3$. Let $X = \begin{bmatrix} x & y & \phi \end{bmatrix}^T \in \mathcal{X} \subset \mathbb{R}^3$ be the entire state of the robot. Then the dynamics can be rewritten as

$$\dot{X} = \begin{bmatrix} cos(\phi) & 0 \\ sin(\phi) & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix} \tag{6.2}$$

Let $g(X) = \begin{bmatrix} cos(\phi) & 0 \\ sin(\phi) & 0 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} R(\phi)\bar{L} \\ e_2^T \end{bmatrix}$ where $R(\phi) \in \mathbb{R}^{2\times 2}$ is the standard rotation matrix, $\bar{L} \in \mathbb{R}^{2\times 2}$ is a diagonal matrix with $(1, 0)$ as its diagonal elements, and $e_2 = \begin{bmatrix} 0 & 1 \end{bmatrix}^T$ is the standard basis vector in the y direction. Define $\bar{x} = \begin{bmatrix} x & y \end{bmatrix}^T \in \mathbb{R}^2$. We define the control that is applied to the robot as $u = \begin{bmatrix} v & \omega \end{bmatrix}^T \in \mathbb{R}^2$. We assume that the linear velocity is bounded as $|v| \leq v_{max}$ where $v_{max} \in \mathbb{R}_{>0}$.

Let $\mathcal{G} = \{X \in \mathcal{X} \mid h(X) \geq 0\}$ be a region of interest which is defined as the super zero level set of a function $h : \mathcal{X} \to \mathbb{R}$. We require that the robot converge to $\mathcal{G}$ in a finite time with a desired heading angle given by $\phi_{term} \in (-\pi, \pi]$. We now formally define the problem statement that is addressed in this chapter.

Problem Statement 4. *Given a differential drive mobile robot with unicycle dynamics as in* (6.1)*, synthesize a finite time control barrier function such that*

1. *A QP based feedback controller with the finite time barrier function as the constraint allows the robot to converge to the desired level set $\mathcal{G}$ with a desired angle of arrival ϕ_{term} within a finite time $T \in (0, \infty)$.*

2. *The QP based controller returns a feasible solution.*

6.2 Weighted Polar L_p Functions

In this section, a weighted polar L_p function introduced in [104] is recapped. A level set of the function is used to analytically represent contours or surfaces of bent objects. The idea is to appropriately use a coordinate transformation between the Cartesian system and the polar system to retrieve the surface equation represented by the weighted L_p norm in each domain.

Consider the coordinate transformation $\Gamma : \mathbb{R}^2 \to \mathbb{R}^2$ as follows,

$$\Gamma(\bar{x}) = \begin{bmatrix} \kappa x \\ \kappa y + 1 \end{bmatrix} \tag{6.3}$$

The polar coordinates of $\bar{x}$ transformed by the mapping Γ are given by:

$$\begin{aligned} \mathcal{R}_\Gamma(\bar{x}) &= \sqrt{(\kappa x)^2 + (\kappa y + 1)^2} \\ \theta_\Gamma(\bar{x}) &= \arctan\left(\frac{\kappa y + 1}{\kappa x}\right) \end{aligned} \tag{6.4}$$

Definition 9 (Weighted Polar $2D$ L_p function (Definition 3 in [104]))**.** *Let $\kappa \neq 0$ and $\sigma = (\sigma_1, \sigma_2)$ be an element-wise positive vector. The σ, κ weighted polar $2D$ L_p function $\Omega : \mathbb{R}^2 \to \mathbb{R}$ is the positive definite function*

$$\Omega(\bar{x}) = \left(\left(\frac{|R_\Gamma(\bar{x}) - 1|}{\sigma_2}\right)^p + \left(\frac{|\theta_\Gamma(\bar{x}) - \theta_0|}{\sigma_1}\right)^p\right)^{\frac{1}{p}} \tag{6.5}$$

where $\theta_0 = sign(\kappa) \cdot \frac{\pi}{2}$, and p is even.

An example plot of Ω is shown in Fig 6.1 where $\sigma = (2, 1)$, $\kappa = 0.3927$ and $p = 10$ are chosen in the example. These values have been chosen arbitrarily for representation purposes. As it is graphically shown in the Fig 6.1, the value function is a positive definite function (Lemma 1 in [104]). Different level sets are shown in Fig 6.1 where the contours projected on the $x - y$ plane correspond to different levels. By using the level set of the function, an analytic contour equation for a bending rectangle is achievable and κ corresponds to the constant curvature of the bending. A detailed derivation of the weighted polar L_p function and its use in optimal path planning can be found in [104]. In this chapter, the terminal level set is defined as the level set of a modified weighted polar L_p function, which has a similar bent rectangular contour as its level set.

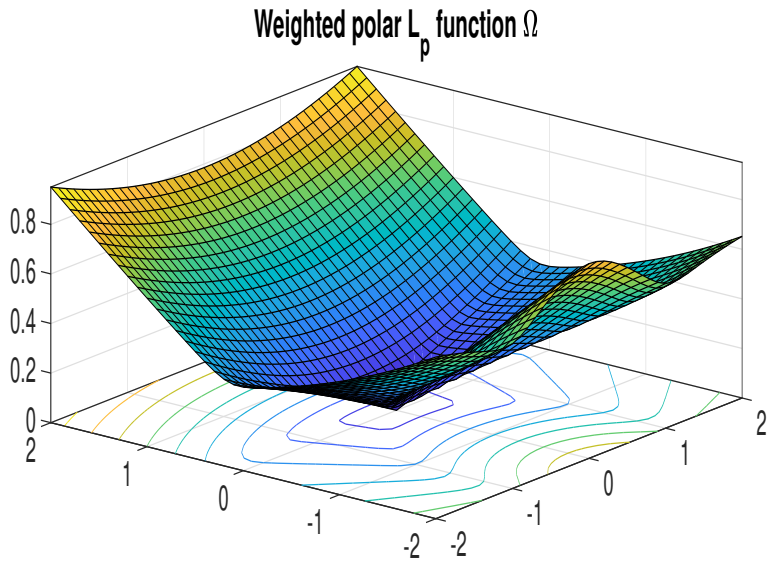


Figure 6.1: The level sets of $\Omega(\bar{x})$ for $p = 10$, $\kappa = 0.3927$, $\sigma = (2, 1)$. Observe that the function is positive definite and $\Omega(\bar{x}) = 0$ at $\bar{x} = \begin{bmatrix} 0 & 0 \end{bmatrix}^T$.

6.3 Near Identity Diffeomorphism

The unicycle kinematics model is nonlinear and nonholonomic. Hence, the robot cannot move sideways instantaneously, but must rotate and then move. The approach adopted in this chapter is to control a virtual point which is not at the center of mass but a point apart from it. The new coordinate which is centered at the virtual point can be obtained via the near identity diffeomorphism discussed in [69]. We control a point at a distance l ahead of the centre of mass and in the direction of the heading. This is as shown in Fig 6.2.

Let $r = \begin{bmatrix} r_x & r_y \end{bmatrix}^T \in \mathbb{R}^2$ be the new virtual point which is controlled. That is,

$$r = \bar{x} + l \cdot R(\phi) \cdot e_1$$

where $R(\phi) = \begin{bmatrix} cos(\phi) & -sin(\phi) \\ sin(\phi) & cos(\phi) \end{bmatrix}$, $l \in \mathbb{R}_{>0}$, and $e_1 = \begin{bmatrix} 1 & 0 \end{bmatrix}^T$ is the standard basis element

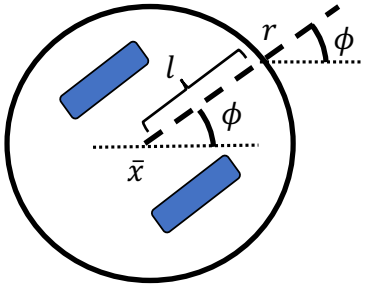


Figure 6.2: The differential drive mobile robot with the centre of mass $\bar{x}$ and the virtual control point r orthogonal to the wheels.

along the x direction. Hence we have

$$r = \bar{x} + l \cdot \begin{bmatrix} cos(\phi) \\ sin(\phi) \end{bmatrix}$$

The dynamics of r are then given by

$$\begin{aligned} \dot{r} &= \dot{\bar{x}} + l \cdot \begin{bmatrix} -sin(\phi) \\ cos(\phi) \end{bmatrix} \cdot \dot{\phi} \\ &= \dot{\bar{x}} + l \cdot R(\phi) \cdot e_2 \cdot \omega \\ &= \begin{bmatrix} vcos(\phi) - lsin(\phi)\omega \\ vsin(\phi) + lcos(\phi)\omega \end{bmatrix}. \end{aligned}$$

Denote the total state of the robot as $\tilde{r} = \begin{bmatrix} r & \phi \end{bmatrix}^T \in \mathcal{X}$. Hence we have

$$\dot{\tilde{r}} = \begin{bmatrix} cos(\phi) & -lsin(\phi) \\ sin(\phi) & lcos(\phi) \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix}. \tag{6.6}$$

Define $L \in \mathbb{R}^{2\times 2}$ to be a diagonal matrix with $(1, l)$ as its diagonal elements.

$$\dot{\tilde{r}} = \begin{bmatrix} R(\phi)L \\ e_2^T \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix} \tag{6.7}$$

Let $\tilde{g}(r) = \begin{bmatrix} R(\phi)L & e_2^T \end{bmatrix}^T$. Moving forward, we will consider this virtual point as the state of the robot and use these dynamics for the analysis of the Lie derivatives and the gradients.

Weighted Polar L_p Barrier Functions

Consider the original state of the robot $X = \begin{bmatrix} x & y & \phi \end{bmatrix}^T$. We consider the following candidate weighted L_p function $\Upsilon : \mathcal{X} \to \mathbb{R}$:

$$\Upsilon(X) = |\kappa| - \left(\left(\frac{R_\Gamma(\bar{x}) - c}{\sigma_2} \right)^p + \left(\frac{\theta_\Gamma(\bar{x}) - \theta_0}{\sigma_1} \right)^p + \mu \cdot \left(\frac{|\kappa| \cdot (\phi - \phi_{term})}{\sigma_3} \right)^p \right)^{\frac{1}{p}} \tag{6.8}$$

where ϕ and ϕ_{term} are wrapped to $(-\pi, \pi]$, p is even and $\mu \in \mathbb{R}_{>0}$ is a design parameter. We will see later that μ influences the error bound on the angle of convergence to the terminal level set.

We define the set of points at which the Lie derivative of (6.8) in the direction of the actuation vector field g vanishes. More formally, we have the following definition.

Definition 10 (Singular set)**.** *A singular set of the system as in* (6.1) *is defined as* $\mathcal{S} = \{X \in \mathcal{X} \mid L_g\Upsilon(X) = 0\}$*, where* $L_g\Upsilon(X) = \frac{\partial \Upsilon(X)}{\partial X} g(X)$ *is the Lie derivative of* (6.8) *along the direction of the actuation vector field* g*.*

Since Υ is smooth in the domain $\mathcal{X}$ and the dimension of $\mathcal{S}$ is strictly less than 3, the singular set is measure zero. As μ is also a control parameter, the singular set can be eliminated by choosing a specific value of μ.

Before considering a proper choice of μ, the following lemma shows that the singular set $\mathcal{S}$ is always nonempty for any $\mu \in \mathbb{R}_{>0}$ if the original state equation in (6.1) is used

instead of (6.7).

Lemma 2. *Consider the weighted polar L_p function as in* (6.8)*. Let $\mathcal{S} = \{X \in \mathcal{X} | L_g\Upsilon(X) = 0\}$ where $L_g\Upsilon(X) = \frac{\partial\Upsilon(X)}{\partial X}g(X)$ is the Lie derivative of Υ along g as in (6.1). Then the set $\mathcal{S}$ is non empty for any $\mu \in \mathbb{R}_{>0}$*

Proof. Define $\alpha(X) = \left(\left(\frac{R_\Gamma(\bar{x})-c}{\sigma_2}\right)^p + \left(\frac{\theta_\Gamma(\bar{x})-\theta_0}{\sigma_1}\right)^p + \mu \cdot \left(\frac{|\kappa|\cdot(\phi-\phi_{term})}{\sigma_3}\right)^p\right) \in \mathbb{R}_{>0}$. The Lie derivative of the weighted L_p function along g is given by:

$$\begin{aligned} L_g\Upsilon(X) &= \frac{\partial\Upsilon(X)}{\partial X}g(X) \\ &= \alpha(X)^{\frac{1-p}{p}}\begin{bmatrix} -\Omega(\bar{x})^{p-1}\nabla\Omega(\bar{x}) & -\frac{\mu|\kappa|^p(\phi-\phi_{term})^{p-1}}{\sigma_3^p}\end{bmatrix}\begin{bmatrix} R(\phi)\bar{L} \\ e_2^T\end{bmatrix}. \end{aligned}$$

Now define $\Psi : \mathbb{R} \to \mathbb{R}$ such that

$$\Psi(\phi) = \frac{\mu|\kappa|^p(\phi-\phi_{term})^{p-1}}{\sigma_3^p}. \tag{6.9}$$

Then we have

$$L_g\Upsilon(X) = \alpha(X)^{\frac{1-p}{p}}\begin{bmatrix} -\Omega(\bar{x})^{p-1}\nabla\Omega(\bar{x}) & -\Psi(\phi)\end{bmatrix}\begin{bmatrix} R(\phi)\bar{L} \\ e_2^T\end{bmatrix}$$

The Lie derivative, $L_g\Upsilon$, at X is given by:

$$L_g\Upsilon(X) = \alpha(X)^{\frac{1-p}{p}}\cdot(-\Omega(\bar{x})^{p-1}\nabla\Omega(\bar{x})R(\phi)\bar{L} - \Psi(\phi)e_2^T)$$

$L_g\Upsilon(X) = 0_{1\times 2}$ implies that $\nabla\Omega(\bar{x})R(\phi)e_1 = 0$ and $\phi = \phi_{term}$ holds since $\bar{L}$ is singular. This condition implies that $L_g\Upsilon(X) = 0_{1\times 2}$ if the heading is equal to the desired heading ϕ_{term}, and at that point, the gradient of Ω is orthogonal to the heading. Since there always exists a point $X \in \mathcal{X}$ such that the gradient $\nabla\Omega(\bar{x})$ is orthogonal to the heading in $\mathcal{X}$, if the

heading of X is equal to ϕ_{term} then $X \in \mathcal{S}$. Hence, there does not exist a $\mu \in \mathbb{R}_{>0}$ such that $\mathcal{S}$ is empty. This concludes the proof. ■

Lemma 1 proves that there exists no value of μ for which the singular set $\mathcal{S}$ as in Definition 10 is empty. This is a problem with regard to QP feasibility because $L_g\Upsilon(X) = 0$ for any $X \in \mathcal{X}$ will cause violation of (1.4) and will render the QP infeasible at X. Hence, the control of the differential drive robot will be considered not with the kinematics in (6.1) but with the NID in (6.7) in order to render $\mathcal{S}$ empty.

Condition for Vanishing Lie Derivative

We observe that (1.4) is the condition that must be satisfied for the function Υ to be a finite time convergence control barrier function. This requires the Lie derivative $L_{\tilde{g}}\Upsilon$ along $\tilde{g}$ to be non zero at each point in the state space. That is, $L_{\tilde{g}}\Upsilon(\tilde{r}) \neq 0$ for all $\tilde{r} \in \mathcal{X}$. The following Lemma characterizes the condition at which the Lie derivative is zero in the state space.

Lemma 3. *Consider the unicycle robot with the near identity diffeomorphism kinematics as in* (6.7)*. The singular set $\mathcal{S}$ in Definition 10 is nonempty only if*

$$\nabla\Omega^T(r) + \frac{\Psi(\phi)R(\phi)e_2}{l\Omega(r)^{p-1}} = 0 \tag{6.10}$$

holds, where $\Psi(\phi) = \frac{\mu|\kappa|^p(\phi-\phi_{term})^{p-1}}{\sigma_3^p}$ for all $\tilde{r} \in \mathcal{X}$.

Proof. The Lie derivative of the weighted polar L_p function along $\tilde{g}$ for all $\tilde{r} \in \mathcal{X}$ is given by

$$\begin{aligned} L_{\tilde{g}}\Upsilon(\tilde{r}) &= \frac{\partial\Upsilon(\tilde{r})}{\partial\tilde{r}}\tilde{g} \\ &= \alpha(\tilde{r})^{\frac{1-p}{p}} \begin{bmatrix} -\Omega(r)^{p-1}\nabla\Omega(r) & -\frac{\mu|\kappa|^p(\phi-\phi_{term})^{p-1}}{\sigma_3^p} \end{bmatrix} \begin{bmatrix} R(\phi)L \\ e_2^T \end{bmatrix} \end{aligned}$$

Hence we have

$$L_{\tilde{g}}\Upsilon(\tilde{r}) = \alpha(r)^{\frac{1-p}{p}}(-\Omega(r)^{p-1}\nabla\Omega(r)R(\phi)L - \Psi(\phi)e_2^T) \tag{6.11}$$

Suppose $L_{\tilde{g}}\Upsilon(\tilde{r}) = 0$. Then we have

$$0 = -\Omega(r)^{p-1}\nabla\Omega(r)R(\phi)L - \Psi(\phi)e_2^T \tag{6.12}$$

Since $R(\phi)L$ is nonsingular,

$$\nabla\Omega(r)^T = -\frac{\Psi(\phi)R(\phi)e_2}{l\Omega(r)^{p-1}} \tag{6.13}$$

holds, and the lemma follows. ■

Observe that the contrapositive of Lemma 3 indicates that if $\langle\nabla\Omega(r), R(\phi)e_1\rangle \neq 0$ or $||\nabla\Omega^T(r)|| \neq \frac{|\Psi(\phi)|}{l\Omega(r)^{p-1}}$, then $||L_{\tilde{g}}\Upsilon(\tilde{r})||$ is not equal to zero at $\tilde{r}$. This means that if we pick the value of the control parameter μ such that $||\nabla\Omega^T(r)|| \neq \frac{|\Psi(\phi)|}{l\Omega(r)^{p-1}}$, then we have $L_{\tilde{g}}\Upsilon(\tilde{r}) \neq 0$ for all $\tilde{r} \in \mathcal{X}$.

6.4 Weighted polar L_p Barrier Functions

Consider the weighted polar L_p function as in (6.8). We prove that it is a valid finite time convergence control barrier function in order to provide a formal guarantee for convergence to the terminal level set at the desired angle. Note that it is sufficient to check if there exists at least one $u = \begin{bmatrix} v & \omega \end{bmatrix}^T \in \mathbb{R}^2$ such that (1.4) is satisfied. To do this, we require a compact domain that encapsulates the zero level set of the weighted polar L_p function. The following theorem provides a characterization of a new domain local to the initial condition $\tilde{r}_0$ of the robot.

Theorem 7. *Let* $\tilde{r}_0 = \begin{bmatrix} r_0 & \phi_0 \end{bmatrix}^T \in \mathcal{X}$ *be the initial configuration of the robot. Define*

$\partial\mathcal{G} = \{\tilde{r} \in \mathcal{X} | \Upsilon(\tilde{r}) = 0\}$. *Then, there exists* $\epsilon_1 > 0$, $\epsilon_2 \in (0, |\kappa|)$, $\mu \in (0, \hat{\mu}_{max}]$, $\mathcal{D} := \mathcal{M} \backslash \mathcal{N}$ *where* $\tilde{r}_0 \in \mathcal{D}$ *and*

$$\mathcal{M} := \{\begin{bmatrix} r & \phi \end{bmatrix}^T \in \mathcal{X} \mid |\kappa| - \Omega(r) \leq \epsilon_2\} \tag{6.14}$$

$$\mathcal{N} := \{\begin{bmatrix} r & \phi \end{bmatrix}^T \in \mathcal{X} \mid |\kappa| - \Omega(r) < -\epsilon_1\}, \tag{6.15}$$

such that $\partial\mathcal{G} \subset \mathcal{D}$.

Proof. We choose $\epsilon_1 > 0$ to be large enough such that $\Upsilon(\tilde{r}_0) \geq -\epsilon_1$. This is possible since $\phi - \phi_{term} \in (-2\pi, 2\pi]$ is bounded. In fact, this ϵ_1 can be chosen such that

$$\sup_{\tilde{r} \in \mathcal{N}^c} \Upsilon(\tilde{r}) \leq -\epsilon_1$$

Observe that this implies that $\Omega(r_0) \geq -\epsilon_1$, and so $\tilde{r}_0 \in \mathcal{N}^c$ where $\mathcal{N}^c$ is the complement of $\mathcal{N}$.

Now, pick $\epsilon_2 \in (0, |\kappa|)$, then

$$\begin{aligned} \Upsilon(\tilde{r}) &= |\kappa| - \left((|\kappa| - \epsilon_2)^p + \mu \cdot \frac{|\kappa|^p \cdot (\phi - \phi_{term})^p}{\sigma_3^p} \right)^{\frac{1}{p}} \\ &\geq |\kappa| - \left((|\kappa| - \epsilon_2)^p + \mu \cdot \frac{|\kappa|^p \cdot (2\pi)^p}{\sigma_3^p} \right)^{\frac{1}{p}} \end{aligned} \tag{6.16}$$

holds for any $\tilde{r} \in \partial M$. Now pick

$$\mu \leq \frac{\bar{\delta} \cdot \sigma_3^p}{(2\pi)^p}$$

for some $\bar{\delta} \in (0, 1)$ defined as

$$\bar{\delta} = 1 - \frac{(|\kappa| - \epsilon_2)^p}{|\kappa|^p}.$$

By substituting μ to (6.16), $\Upsilon(\tilde{r}) \geq 0$ holds for all $\tilde{r} \in \partial\mathcal{M}$.

Define

$$\hat{\mu}_{max} = \frac{\delta \cdot \sigma_3^p}{(2\pi)^p} \tag{6.17}$$

Then from the choice of $\epsilon_1 > 0$, $\epsilon_2 \in (0, |\kappa|)$ and $\mu \in (0, \hat{\mu}_{max})$, the supremum of Υ on N^c is strictly negative, and the infimum of Υ on $\partial\mathcal{M}$ is non-negative. It is also true that

$$\inf_{\tilde{r} \in \mathcal{M}^c} \Upsilon(\tilde{r}) > 0,$$

and therefore, the intersection $\partial G \cap \mathcal{D}^c$ is empty. Hence $\partial G \subset \mathcal{D}$. ■

Now, having clearly defined the domain, we prove that there is no singular set (Definition 10) within the domain $\mathcal{D}$ for a fixed μ which uses Theorem 7.

Theorem 8. *There exists a uniform upper bound $\mu_{max} > 0$, such that if $\mu \in (0, \mu_{max}]$, then $||L_{\tilde{g}} \Upsilon(\tilde{r})|| > 0$ for all $\tilde{r} \in \mathcal{D}$.*

Proof. Observe that $\mathcal{D}$ is compact in $\mathcal{X}$ since $\mathcal{N}$ is open, and Υ is continuously differentiable in $\mathcal{D}$. Therefore, there exists a minimum,

$$K := \min_{\tilde{r} \in \mathcal{D}} ||\nabla \Omega(r)||$$

and,

$$\Omega_{min} := \min_{\tilde{r} \in \mathcal{D}} \Omega(r)$$

Note that $\Omega(r) \neq 0$ for all $\tilde{r} \in \mathcal{D}$ since Ω is positive definite and $\begin{bmatrix} 0 & 0 \end{bmatrix}^T \notin \mathcal{D}$. Now, pick $\mu^* > 0$ such that

$$\frac{\mu^* |\kappa|^p (2\pi)^{p-1}}{l \sigma_3^p \Omega_{min}^{p-1}} \leq \frac{K}{\beta}, \text{ where } \beta > 1 \tag{6.18}$$

Now suppose that if $\tilde{r} \in \mathcal{D}$, then by its definition in (6.9), $\Psi(\phi) = 0$ if and only if $\phi = \phi_{term}$ since $\mu > 0$. Observe that if $\Psi(\phi) = 0$, then $||L_{\tilde{g}} \Upsilon(\tilde{r})|| \neq 0$ since $||\nabla \Omega(r)|| \neq 0$.

And so it is enough to prove that $\|L_{\tilde{g}}\Upsilon(\tilde{r})\| \neq 0$ for $\phi \neq \phi_{term}$.

Since $\phi \in (-\pi, \pi]$, then for all $\tilde{r} \in \mathcal{D}$, the following holds,

$$\frac{(\phi - \phi_{term})^{p-1}}{\sigma_3^p \Omega(r)^{p-1}} \leq \frac{(2\pi)^{p-1}}{\sigma_3^p \Omega_{min}^{p-1}}$$

since $p > 1$. This implies that

$$\left\| \frac{\Psi(\phi)R(\phi)e_2}{l\Omega(r)^{p-1}} \right\| \leq \frac{\mu|\kappa|^p \cdot (2\pi)^{p-1}}{l\sigma_3^p \Omega_{min}^{p-1}}$$

since $R(\phi)$ is a unitary matrix. Now, pick

$$\mu < \min\{\mu^*, \hat{\mu}_{max}\}, \tag{6.19}$$

where $\hat{\mu}_{max}$ is as in Lemma 3. Then for all $\tilde{r} \in \mathcal{D}$ we have

$$\left\| \frac{\Psi(\phi)R(\phi)e_2}{l\Omega(r)^{p-1}} \right\| \leq \frac{K}{\beta} < \|\nabla\Omega(r)\|$$

since $\beta > 1$. By invoking the contrapositive of Lemma 3, $\|L_{\tilde{g}}\Upsilon(\tilde{r})\| \neq 0$ holds for all $\tilde{r} \in \mathcal{D}$. Hence, the upper bound $\min\{\mu^*, \hat{\mu}_{max}\}$ suffices the requirement. ■

Next, we show that the weighted polar L_p function is indeed a valid finite time control barrier function in $\mathcal{D}$.

Theorem 9. *In the domain $\mathcal{D}$ with the desired level set $\mathcal{G} = \{\tilde{r} \in \mathcal{X} | \Upsilon(\tilde{r}) \geq 0\}$, the weighted polar L_p function Υ renders* (1.4) *true for all $\tilde{r} \in \mathcal{D}$ and there exists a finite time $0 < T < \infty$ such that $\Upsilon(\tilde{r}(T)) \in \mathcal{G}$.*

Proof. Let $\mathcal{D}$ be the domain as defined in Theorem 7. Pick μ as proposed in Theorem 8. Then, from Theorem 8, we have $L_{\tilde{g}}\Upsilon(\tilde{r}) \neq 0$ for all $\tilde{r} \in \mathcal{D}$. Now for all $\tilde{r} \in \mathcal{D}$, pick

$$u = \frac{L_{\tilde{g}}\Upsilon(\tilde{r}) \cdot (-\gamma \cdot sign(\Upsilon(\tilde{r})) \cdot |\Upsilon(\tilde{r})|^\rho)}{\|L_{\tilde{g}}\Upsilon(\tilde{r})\|_2^2} \in \mathbb{R}^2 \tag{6.20}$$

This gives us $L_{\tilde{g}}\Upsilon(\tilde{r})u + \gamma \cdot sign(\Upsilon(\tilde{r})) \cdot |\Upsilon(\tilde{r})|^{\rho} = 0$. Hence, for all $\tilde{r} \in \mathcal{D}$, there exists a $u \in \mathbb{R}^2$ such that (1.4) is satisfied. Thus in the domain $\mathcal{D}$, Υ is a valid finite time control barrier function. Note that since we wrap ϕ, ϕ_{term} in (6.8), $\tilde{r}$ remains in $\mathcal{D}$. Since $\partial\mathcal{G} \subset \mathcal{D}$, invoking Proposition 2, and the fact that Υ is positive definite, there exists $T \in (0, \infty)$ such that $\tilde{r}(t) \in \mathcal{D}$ for all $t \in [0, T)$ and $\Upsilon(\tilde{r}(T)) \in \mathcal{G}$. ■

6.5 Bound on Angle of Convergence

Since we are interested in the robot converging to the desired level set with the desired heading angle, it is imperative that the error between the final heading and the desired heading be minimum. This error can be controlled by the design parameter μ. This is formalized in the following proposition.

Proposition 4. *Given a domain $\mathcal{D}$ as defined in Theorem 7 and a desired level set $\mathcal{G} = \{\tilde{r} \in \mathcal{X} | \Upsilon(\tilde{r}) \geq 0\} \subset \mathcal{D}$ with Υ as the associated weighted polar L_p barrier function, any continuous controller $u : \mathcal{D} \to \mathbb{R}^2$ such that $u(\tilde{r}) \in \mathcal{U}$ as in* (1.5)*, drives the robot to $\mathcal{G}$ in a finite time $0 < T < \infty$. Moreover, the angle of convergence of the robot to the desired level set is bounded by*

$$\phi_{term} - \frac{\sigma_3}{\mu^{1/p}} \leq \phi(T) \leq \phi_{term} + \frac{\sigma_3}{\mu^{1/p}} \tag{6.21}$$

where $\phi_{term} \in (-\pi, \pi]$ is the desired heading angle.

Proof. Since Υ is a finite time control barrier function, from Proposition 2, the robot converges to $\mathcal{G}$ in a finite time $0 < T < \infty$. That is, we have $\Upsilon(\tilde{r}) \geq 0$ for all $t \geq T$. Since the robot is also within the $2D$ level set $\Omega(r)$ we have $\Omega(r) = m \cdot |\kappa|$ where $0 < m \leq 1$. That is,

we have

$$
\begin{aligned}
& 0 \leq |\kappa| - \left(m^p \cdot |\kappa|^p + \frac{\mu \cdot |\kappa|^p}{\sigma_3^p} \cdot (\phi(T) - \phi_{term})^p \right)^{\frac{1}{p}} \\
\iff \quad & m^p \cdot |\kappa|^p + \frac{\mu \cdot |\kappa|^p}{\sigma_3^p} \cdot (\phi(T) - \phi_{term})^p \leq |\kappa|^p \\
\iff \quad & m^p + \frac{\mu}{\sigma_3^p} \cdot (\phi(T) - \phi_{term})^p \leq 1 \\
\iff \quad & \frac{\mu}{\sigma_3^p} \cdot (\phi(T) - \phi_{term})^p \leq (1 - m^p)
\end{aligned}
$$

Let $\xi = 1 - m^p$. Note that $\xi < 1$. Hence we have

$$
(\phi(T) - \phi_{term})^p \leq \frac{\xi \cdot \sigma_3^p}{\mu} \iff |\phi(T) - \phi_{term}| \leq \frac{\xi' \cdot \sigma_3}{\mu^{\frac{1}{p}}}
$$

where $\xi' = \xi^{1/p}$. Since $\frac{\xi' \cdot \sigma_3}{\mu^{\frac{1}{p}}} \leq \frac{\sigma_3}{\mu^{\frac{1}{p}}}$ we have the inequality

$$
\phi_{term} - \frac{\sigma_3}{\mu^{1/p}} \leq \phi(T) \leq \phi_{term} + \frac{\sigma_3}{\mu^{1/p}}
$$

and hence the theorem holds. ■

Note that by setting μ in (6.8) to be large, we can obtain a final terminal angle which is almost equal to the desired terminal angle as long as $||L_{\tilde{g}} \Upsilon(\tilde{r})|| \neq 0$ for all $\tilde{r} \in \mathcal{D}$. The next section details the QP based controller which uses the weighted polar L_p barrier function as the constraint. In particular, this QP based controller returns a feasible solution over all points in the domain $\mathcal{D}$.

Quadratic Program Based Controller Synthesis

We adopt a QP based controller which incorporates the weighted polar L_p finite time barrier function as a constraint, and returns the minimum energy control law point wise in the state

space. In particular, the optimization problem that is solved is given by

$$\begin{aligned} &\min_{u\in\mathbb{R}^2} \quad \|u\|_2^2 \\ &\text{s.t} \quad \frac{\partial \Upsilon(\tilde{r})}{\partial \tilde{r}}\dot{\tilde{r}} \geq -\gamma \cdot sign(\Upsilon(\tilde{r})) \cdot |\Upsilon(\tilde{r})|^{\rho} \end{aligned} \tag{6.22}$$

where $\gamma > 0$, $\rho \in [0, 1)$, and $\Upsilon : \mathcal{D} \to \mathbb{R}$ is the weighted polar L_p barrier function of the form

$$\begin{aligned} \Upsilon(\tilde{r}) = |\kappa| - \Bigg(\mu_1 \cdot \left(\frac{R_\Gamma(r) - c}{\sigma_2}\right)^p + \mu_1 \cdot \left(\frac{\theta_\Gamma(r) - \theta_0}{\sigma_1}\right)^p + \\ \mu_2 \cdot \left(\frac{|\kappa| \cdot (\phi - \phi_{term})}{\sigma_3}\right)^p\Bigg)^{\frac{1}{p}} \end{aligned} \tag{6.23}$$

where $\mu_1 \in \mathbb{R}_{\geq 0}, \mu_2 \in \mathbb{R}_{>0}$ are design parameters.

Since the primary aim of the proposed work is to converge to the desired level set at a desired heading angle, we choose $\mu_1 = 1$ and $\mu_2 \in \mathbb{R}_{>0}$ to be a large value so that the error bound is small as per Proposition 4. However, since this choice of μ_2 could potentially violate (6.19), there is a possibility that the robot enters the singular set $\mathcal{S}$ as defined in Definition 10. The QP will then be infeasible. Therefore, we propose Algorithm 1 which guarantees feasibility of the QP as well as small value of error on the final heading angle. The algorithm implements a two stage controller preemptively when $\|L_{\tilde{g}}\Upsilon(\tilde{r})\| < \delta$ where $\delta \in \mathbb{R}_{>0}$ is a small number. We define this as a deadlock situation. At $t = 0$, DeadLock= 0. The following subsections describe each stage in more detail.

Algorithm 6 QP Based Controller

Input : $L_{\tilde{g}}\Upsilon(\tilde{r})$
Output: $u(\tilde{r})$

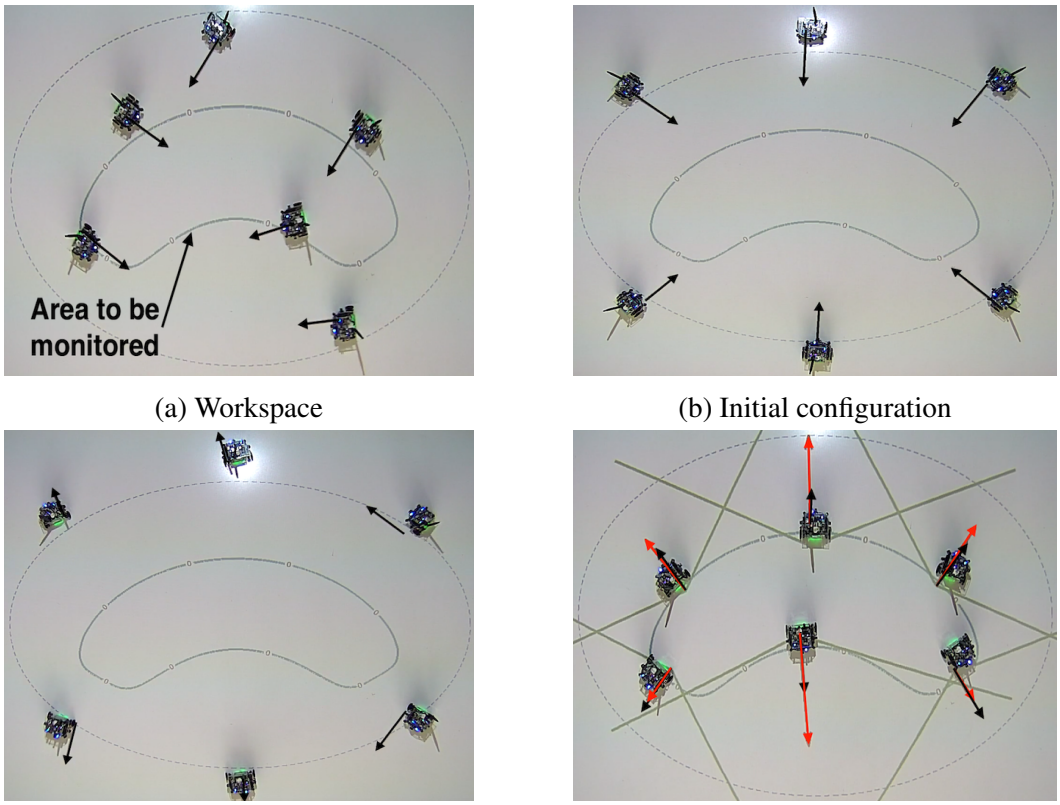
if DeadLock = 1 **then**
  Pick $\mu_1 = 1$, μ_2 such that (6.19) holds
  while $\Upsilon(\tilde{r}) < 0$ **do**
    Solve the QP as in (6.22)
  end while
  Pick $\mu_1 = 0$, $\mu_2 > 0$ to be large
  while $\Upsilon(\tilde{r}) < 0$ **do**
    Solve the QP as in (6.22)
  end while
else
  Pick $\mu_1 = 1$, $\mu_2 > 0$ to be large
  Solve the QP as in (6.22)
  if $\|L_g\Upsilon(\tilde{r})\| < \delta$ **then**
    DeadLock $\leftarrow$ 1
  else
    DeadLock $\leftarrow$ 0
  end if
end if

Stage 1: Convergence to the desired level set

In this stage, we choose $\mu_1 = 1$ and μ_2 as in (6.19). Then for all $\tilde{r} \in \mathcal{D}$, the weighted polar L_p barrier function is given as

$$\Upsilon(\tilde{r}) = |\kappa| - \left(\left(\frac{R_\Gamma(r) - c}{\sigma_2}\right)^p + \left(\frac{\theta_\Gamma(r) - \theta_0}{\sigma_1}\right)^p + \mu_2 \cdot \left(\frac{|\kappa| \cdot (\phi - \phi_{term})}{\sigma_3}\right)^p\right)^{\frac{1}{p}} \tag{6.24}$$

Then, the QP as in (6.22) is solved in order to converge to the desired level set within a finite time $0 < T_1 < \infty$. This signifies the end of stage 1. The next goal is to converge to the desired heading angle. This is discussed in stage 2.



(a) Workspace

(b) Initial configuration

(c) Intermediate configuration

(d) Final configuration

Figure 6.3: In (b), the robots initialize around the target level set described by (6.8) with the parameters in the case study. Using the QP based controller in Algorithm 1, the robots converge to the terminal level set in the direction of the gradient of the level set as seen in (d).

Stage 2: Convergence to the desired heading angle

As mentioned before, the value of μ_2 as in (6.19) is small and hence the bound on the error on the terminal angle of convergence as per Theorem 2 is large. Since we require the robot to be aligned towards the desired heading angle, we choose $\mu_1 = 0$ and $\mu_2 = \mu_{large} > 0$ (a large value). Then for all $\tilde{r} \in \mathcal{D}$, the weighted polar L_p barrier functions is given as

$$\Upsilon(\tilde{r}) = |\kappa| - \left(\mu_{large} \cdot \left(\frac{|\kappa| \cdot (\phi - \phi_{term})}{\sigma_3} \right)^p \right)^{\frac{1}{p}} \tag{6.25}$$

The fact that (6.25) is a valid barrier function is proved in Proposition 5. Again, the QP as in (6.22) is solved and the robot converges to the desired terminal angle in a finite time

$0 < T_2 < \infty$ within an error formalized in Proposition 5.

Proposition 5. *Given a desired terminal angle $\phi_{term} \in (-\pi, \pi]$, $\mu_1 = 0$, $\mu_2 = \mu_{large} > 0$ with Υ as the weighted polar L_p barrier function as in* (6.25)*, any continuous controller $u : \mathcal{D} \to \mathbb{R}^2$ such that $u(\tilde{r}) \in \mathcal{U}$ as in* (1.5)*, drives the robot to a final heading angle given by*

$$\phi_{term} - \frac{\sigma_3}{\mu_{large}^{1/p}} \leq \phi(T) \leq \phi_{term} + \frac{\sigma_3}{\mu_{large}^{1/p}} \tag{6.26}$$

within a finite time $0 < T < \infty$.

Proof. For all $\tilde{r} \in \mathcal{D}$, observe that if $\mu_1 = 0$, then from (6.11) we have $L_{\tilde{g}}\Upsilon(\tilde{r}) \neq 0$ for all $\phi \neq \phi_{term}$. Similar to the argument in Theorem 9, Υ is a finite time barrier function. Hence, $\Upsilon(\tilde{r}) \geq 0$ for all $\tilde{r} \in \mathcal{D}$, $t \geq T$ where $0 < T < \infty$ which yields

$$\begin{aligned}
& 0 \leq |\kappa| - \left(\frac{\mu_{large} \cdot |\kappa|^p}{\sigma_3^p} \cdot (\phi(T) - \phi_{term})^p \right)^{\frac{1}{p}} \\
\iff \quad & \frac{\mu_{large}}{\sigma_3^p} \cdot (\phi(T) - \phi_{term})^p \leq 1 \\
\iff \quad & (\phi(T) - \phi_{term})^p \leq \frac{\sigma_3^p}{\mu_{large}} \\
\iff \quad & |\phi(T) - \phi_{term}| \leq \frac{\sigma_3}{\mu_{large}^{1/p}}
\end{aligned}$$

Hence we have the inequality

$$\phi_{term} - \frac{\sigma_3}{\mu_{large}^{1/p}} \leq \phi(T) \leq \phi_{term} + \frac{\sigma_3}{\mu_{large}^{1/p}}$$

and thus the theorem holds. ∎

6.6 Implementation Results

Consider a team of 6 differential drive mobile robots indexed by the set $\mathcal{I} = \{1, 2, \ldots, 6\}$, and dynamics as in (6.1). Since we use the NID, we consider the virtual point $\tilde{r}_i \in \mathcal{X}$ for all $i \in \mathcal{I}$ where $\tilde{r}_i$ is the total state of agent i and $r_i = \begin{bmatrix} r_x^i & r_y^i \end{bmatrix}^T$. Let $\mathcal{G} = \{\tilde{r} \in \mathbb{R}^3 | \Upsilon(\tilde{r}) \geq 0\}$

for all $i \in \mathcal{I}$ be the terminal level set defined as the super level set of the weighted polar L_p barrier function parametrized by $\sigma = (0.7, 0.2, \frac{\pi}{20})$, $\mu = 100$, $\kappa = 1.220$, and R_Γ, θ_Γ are as in (6.4), $\phi_{i,term}$ is the gradient direction for all $i \in \mathcal{I}$.

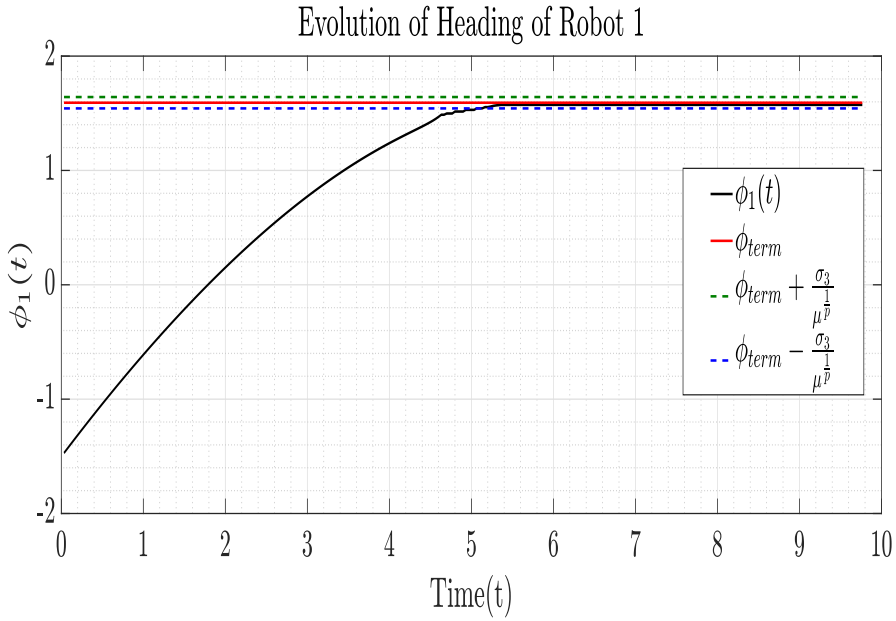


Figure 6.4: The heading of agent 1 converges within the error bound as in Proposition 4

We require the agents to protect an asset which lies within the terminal level set described by (6.8) as shown in Fig 6.3a. The agents are equipped with wide field of view (FOV) sensors. The agents must converge to the terminal level set and align themselves in the direction of the gradient vector to the level set so that any incoming threat is detected by at least one agent. Using expressions for the gradient computation in [104], we determine the gradient vector at each point in the state space, and that is the desired terminal angle $\phi_{i,term}$. At the same time, the agents must also avoid collisions with each other. The collision avoidance barrier certificate is given by

$$h_{safety}(r_i) = ||r_i - r_j||_2^2 - d_{safe}^2 \text{ for all } i \in \mathcal{I}, j \in \mathcal{N}_i$$

where $d_{safe} \in \mathbb{R}_{>0}$ is a safety radius and $\mathcal{N}_i$ is the set of all agents that lie within the

neighborhood of agent i.

By solving a decentralized QP as in (6.22) with the additional collision avoidance barrier functions online and point wise, the agents converge to the terminal level set in the direction of the gradient as shown in Fig. 3. The red arrows indicate the desired heading, the black arrows indicate the final heading of the robots, and the green lines are the FOV for the robots. Observe that no intrusion is possible since an intruder will be detected by at least one agent. We chose $\mu = 100$ which is large. Hence, from Proposition 4, the error on the final angle of convergence is also very small. Fig. 4. shows the evolution of the heading for agent 1. Observe that the heading converges to within the bound as per Proposition 4. No deadlocks were detected since (6.13) was always violated. We provide a video of the experiment (`https://youtu.be/oTLoJNgs3bo`) conducted on the Robotarium multi-robot testbed at Georgia Tech [67].

6.7 Concluding Remarks

In this chapter, we introduced a new finite time control barrier function inspired by the weighted L_p norm. This function encodes the full state of the unicycle, as opposed to existing methods in literature. We also proved that existing methods cannot be used directly since there exist singular sets in the state space. We characterized the condition for the singular sets and determined the expression for the control parameters in order to eliminate these sets. We proposed a QP based controller which returns feasible solutions. We provided a multi-agent surveillance case study in addition to the simulation and experimental results.

Part III

Barrier Functions for Volume

CHAPTER 7

EXTENT-COMPATIBLE CONTROL BARRIER FUNCTIONS

In the previous two chapters, we discussed techniques for guaranteeing feasibility for the quadratic program-based controller. This involved composing multiple barrier functions, relaxing the constraints in the program using slack variables, and leveraging the structure of a special class of barrier functions. In this chapter, we focus on a very practical limitation of traditional barrier function based methods.

Traditional CBF based approaches ignore the physical dimensions of the system and the emphasis is on the forward invariance of a single point defining the system state. The volume (or extent) of the system is not explicitly incorporated in the formulation of the constraints in the QP. In this chapter, we propose a CBF based method for imposing safety constraints on a control-affine dynamical system with extent. We guarantee safety of the system using a modified CBF constraint which ensures that the extent set always remains within the safe set. We first propose implementing the resulting constraint using a sum-of-squares (SOS) optimization program [107]. Since SOS programs can be computationally difficult for high dimensional systems and are only applicable when the safe and extent sets can be represented as polynomials, we next prove that the guarantee on system safety can be retained by considering only a finite set of sampled points on the boundary of the extent set, and we propose a QP based controller using the sampled points. Lastly, the proposed framework is validated both in simulations as well as on a robotic platform.

We note that, in principle, a system's extent can instead be accommodated by shrinking the safe set appropriately [108]. However, CBF-based methods rely on characterizing the safe set as a level-set of a function that is known in closed form. Obtaining such a closed form function to accommodate a system's extent is generally not possible. An important exception is when the system's extent and safe set can be represented as balls in the same

normed space. This feature is implicitly exploited in, e.g., [109], which proposes a CBF based method to avoid collisions between teams of quadrotors. This method requires the volume of each agent to be identical and collision avoidance is then achieved by virtue of the fact that every agent's volume corresponds to identical balls in a normed space. Here, we propose a method that does not require such restrictive assumptions. Compared to the very recent work in sensor based navigation [110] which only considers strongly convex unsafe sets, our method can be extended to account for non-convex sets as well by using recent results by [111]. The latest work by [112] uses a hybrid controller and a computationally expensive partition of the configuration space of the robot. This is avoided in our work since barrier function based techniques do not require a partition of the configuration space.

7.1 Problem Statement

Given a ZCBF, [73, Theorem 6] guarantees that the system state will remain within the safe set $\mathcal{C}$ when control inputs are chosen according to $u(x) \in U(x)$ for all $x \in \mathcal{D}$. This notion of safety, however, disregards the extent of the system. To that end, we define a notion of system safety that includes the physical volume of the system. We encapsulate this notion of volume with an extent function.

Definition 11 (Extent Function)**.** *An* extent function $E : \mathcal{D} \times \mathcal{D} \to \mathbb{R}$ *is a continuously differentiable function such that:*

1. $\mathcal{E}(x) = \{y \in \mathcal{D} \mid E(x, y) \leq 0\}$ *is nonempty for all* $x \in \mathcal{D}$*,*
2. $\frac{\partial}{\partial y} E(x, y) \neq 0$ *for all* (x, y) *such that* $E(x, y) = 0$*, and*
3. *for all* $\delta > 0$*, there exists* $\epsilon > 0$ *such that for all* $x \in \mathcal{D}$*, if* $E(x, y) \leq \epsilon$ *then* $\inf_{\widehat{y}: E(x,\widehat{y})=0} \|y - \widehat{y}\| \leq \delta$.

Condition 2 is essentially a regularity condition on the extent function E that prevents it from becoming "flat" at the boundary. This is analogous to the regularity assumption on

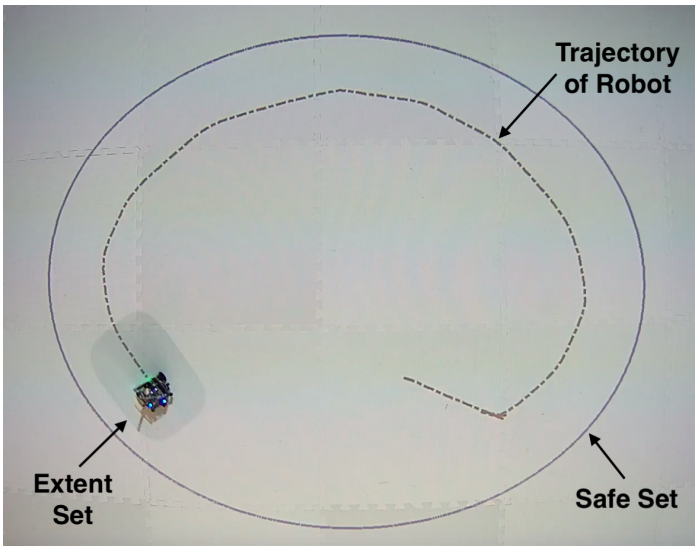


Figure 7.1: A motivating example, where a differential drive robot with a system volume covered by an extent set must stay within the ellipsoidal safe set when a nominal unsafe controller tries to drive it outside. This chapter proposes controller frameworks to guarantee safety of the system including its volume under such situations. The image is from a Robotarium [67] implementation of the framework introduced in this chapter, which we also describe in Section 8.5.

h as is made in, e.g., [113]. Technically, this type of regularity condition is important for barrier-based methods since such methods rely on certain gradient functions being nonzero. Given an extent function, the set $\mathcal{E}(x)$ above defines the extent of the system when the state of the system is $x \in \mathcal{D}$, and $\partial\mathcal{E}(x) = \{y \in \mathcal{D} \mid E(x, y) = 0\}$ denotes the extent boundary. We aim to ensure that the extent of the system is contained within the safe set for all time, i.e., $\mathcal{E}(x(t)) \subseteq \mathcal{C}$ for all $t \geq 0$ along trajectories of (1.1). An example of such a problem setup is shown in Fig. 7.1.

Problem Statement 5. *Given a control affine dynamical system as in* (1.1) *with extent function* $E(x, y)$*, synthesize a controller which guarantees* $\mathcal{E}(x(t)) \subseteq \mathcal{C}$ *for all* $t \geq 0$ *whenever* $\mathcal{E}(x(0)) \subset \mathcal{C}$.

7.2 Extent-Compatible Control Barrier Functions (Ec-CBF)

Given a system of the form (1.1) we now introduce extent-compatible control barrier functions (Ec-CBFs) which are analogous to ZCBFs, but guarantee that the entire extent set remains within the safe set under the action of a suitable control input.

Definition 12 (Extent-Compatible Control Barrier Function (Ec-CBF))**.** *A continuously differentiable function h is an* extent-compatible control barrier function *(Ec-CBF) for the system* (1.1) *with extent function E if $\frac{\partial}{\partial x}h(x) \neq 0$ for all x such that $h(x) = 0$ and there exists locally Lipschitz class $\mathcal{K}$ functions α_1 and α_2 such that for all $x \in \mathcal{D}$ with $\mathcal{E}(x) \subseteq \mathcal{C}$ and for all $y \in \mathcal{C}$, defining*

$$\mathcal{M}(x,y,u) \quad := \quad \frac{\partial E(x,y)}{\partial x}(f(x) \;+\; g(x)u) \;+\; \alpha_1(E(x,y)) \;+\; \alpha_2(h(y))\,,$$

it holds that $\sup_{u\in\mathbb{R}^m}\{\mathcal{M}(x,y,u)\} \geq 0$.

Given an Ec-CBF h, for all $x \in \mathcal{D}$, define the set

$$\mathcal{U}(x) = \{u \in \mathbb{R}^m \mid \mathcal{M}(x,y,u) \geq 0 \text{ for all } y \in \mathcal{C}\}\,. \tag{7.1}$$

Assuming that the extent of the system initially begins inside the safe region, choosing a control input $u(x) \in \mathcal{U}(x)$ at any given state $x \in \mathcal{D}$ guarantees that $\mathcal{E}(x(t)) \subseteq \mathcal{C}$ for all $t \geq 0$, as formalized in the following theorem.

Theorem 10. *Consider system* (1.1) *with initial state $x(0)$, an extent function E, and an Ec-CBF h with associated safe set $\mathcal{C} = \{x \in \mathcal{D} \mid h(x) \geq 0\} \subset \mathcal{D}$. If $\mathcal{E}(x(0)) \subset \mathcal{C}$, then any continuous feedback controller $u : \mathcal{D} \to \mathbb{R}^m$ such that $u(x) \in \mathcal{U}(x)$ for all $x \in \mathcal{D}$ guarantees that $\mathcal{E}(x(t)) \subseteq \mathcal{C}$ for all $t \geq 0$.*

Proof. Suppose by contradiction that the assumptions of the theorem hold but there exists a time $t' > 0$ such that $\mathcal{E}(x(t')) \notin \mathcal{C}$, that is, the system is unsafe at time t'. This means

there exists a point $y' \in \mathcal{D}$ such that $E(x(t'), y') \leq 0$ and $h(y') < 0$. In fact, we can further assume $E(x(t'), y') < 0$. Indeed, if $E(x(t'), y') = 0$, then because $\frac{\partial}{\partial y} E(x(t'), y') \neq 0$ by the definition of extent function, for small enough ϵ, it holds that $E(x(t'), y'') < 0$ and $h(y'') < 0$ where $y'' = y' - \epsilon \frac{\partial}{\partial y} E(x(t'), y')$, and we could consider y'' instead of y'. Since the system is assumed to be initialized in safe conditions with $\mathcal{E}(x(0)) \subset \mathcal{C}$, and since trajectories are continuous, there must exist an earliest time t''' and a point y''' when the system just becomes unsafe, *i.e.*, there exists $0 < t''' \leq t'$ and $y''' \in \mathcal{D}$ such that $h(y''') = 0$ and $E(x(t'''), y''') < 0$.

Let $w(t) = E(x(t), y''')$, *i.e.*, $w(t)$ is the value of the extent function at the point y''' over time. Since the system is assumed to be initially safe, it holds that $w(0) = E(x(0), y''') \geq 0$, and by construction, $w(t''') = E(x(t'''), y''') < 0$. But, for all $t \geq 0$,

$$\begin{aligned} \dot{w}(t) &= \frac{\partial E(x(t), y''')}{\partial x} \big(f(x(t)) + g(x(t)) u(x(t)) \big) \\ &\geq -\alpha_1(w(t)) - \alpha_2(h(y''')) \\ &= -\alpha_1(w(t)) \,, \end{aligned}$$

where the first inequality holds since h is an Ec-CBF and the second equality follows because $-\alpha_2(h(y''')) = 0$.

Now, consider the initial value problem $\dot{\eta}(t) = -\alpha_1(\eta(t))$ with $\eta(0) = w(0)$. Note that $\eta(0) \geq 0$ since $w(0) \geq 0$. The comparison lemma [7, Lemma 3.4] then implies $w(t) \geq \eta(t) \geq 0$ for all $t \geq 0$. But this contradicts that $w(t''') < 0$. Hence, $\mathcal{E}(x(t)) \subseteq \mathcal{C}$ for all $t \geq 0$. ■

Observe that the only feature of α_2 used in the proof of Theorem 10 is that $\alpha_2(0) = 0$. The additional properties on α_2 imposed in Definition 12 are useful for practical implementation and exploited in Theorem 12.

Remark 5. *From the proof of Theorem 10, we observe that* $\sup_u \{\mathcal{M}(x, y, u)\} \geq 0$ *from Definition 12 only has to hold for* $y \in \mathcal{D}$ *such that* $E(x, y)$ *is in a neighborhood of* 0*. Part*

3) in Definition 11 ensures that for a small enough neighborhood, such y must be arbitrarily close to the boundary $\partial\mathcal{E}(x)$.

7.3 Minimally Invasive Quadratic Program Controller

In the scenario where a system designer would like to employ some unsafe nominal feedback control policy $k : \mathcal{D} \to \mathbb{R}^m$ on the system (1.1) with extent, we propose incorporating (7.1) as a constraint at runtime to obtain a minimally invasive quadratic program (QP) based controller using a Ec-CBF, similar to the technique proposed in [19] for ZCBFs. In particular, we propose a quadratic program based controller of the form

$$u_{\text{QP}}(x) = \underset{u \in \mathcal{U}(x)}{\arg\min} \ \|u - k(x)\|_2^2 . \tag{7.2}$$

For fixed x, $\mathcal{U}(x)$ is defined from (7.1) and requires a given inequality to hold for all y, leading to an infinite number of linear constraints on u. In the remainder of this section, we present two approaches to obtaining computationally tractable programs that retain safety guarantees.

7.3.1 Optimization Over Sum-of-Squares Polynomials

We recast (7.2) as a sum-of-squares (SOS) optimization problem in the independent variable y. Recall that x is the state of the system and y is any point in the domain $\mathcal{D}$.

Definition 13 (Sum-of-Squares (SOS) Polynomials)**.** *A polynomial $s(y)$ is a* sum-of-squares *polynomial if it can be written as $s(y) = \sum_{i=1}^{\ell} p_i(y)^2$ for some natural number ℓ where each $p_i(y)$ is a polynomial. Let $\Sigma[y]$ denote the set of all SOS polynomials. Note that if $s(y) \in \Sigma[y]$, then $s(y) \geq 0$ for all $y \in \mathbb{R}^n$.*

Theorem 11. *Consider system* (1.1) *with initial state $x(0)$, an extent function E, and an Ec-CBF h with associated safe set $\mathcal{C} = \{x \in \mathcal{D} \mid h(x) \geq 0\} \subset \mathcal{D}$. Further assume $E(x, y)$ and $\alpha_1(E(x, y))$ are polynomial in y for any fixed x and that $h(y)$ and $\alpha_2(h(y))$ are*

polynomial in y. Let $k : \mathcal{D} \to \mathbb{R}$ be a continuous nominal controller and suppose $\mathcal{E}(x(0)) \subset \mathcal{C}$. If the set

$$\widetilde{\mathcal{U}}(x) = \left\{ u \in \mathbb{R}^m \;\middle|\; \frac{\partial E(x,y)}{\partial x}(f(x) + g(x)u) + \alpha_1(E(x,y)) + \alpha_2(h(y)) - s(y)h(y) \in \Sigma[y] \right. \\ \left. \text{for some } s(y) \in \Sigma[y] \right\}$$

is non-empty for all $x \in \mathcal{D}$, then the solution $x(t)$ of system (1.1) *with*

$$u(x) = u_{SOS}(x) := \underset{u \in \widetilde{\mathcal{U}}(x)}{\arg\min} \|u - k(x)\|_2^2 \tag{7.3}$$

is such that $\mathcal{E}(x(t)) \subseteq \mathcal{C}$ for all $t \geq 0$.

Proof. For each x, the optimization problem (7.3) is feasible by hypothesis, and the fact that $u \in \widetilde{\mathcal{U}}(x)$ implies

$$\frac{\partial E(x,y)}{\partial x}(f(x) + g(x)u) + \alpha_1(E(x,y)) + \alpha_2(h(y)) - s(y)h(y) \geq 0$$

for all $y \in \mathbb{R}^n$, since the left hand side of the inequality is required to be an SOS polynomial. We know that $s(y)$ is an SOS polynomial and hence $s(y) \geq 0$ for all $y \in \mathbb{R}^n$. Next, observe that $s(y)h(y) \geq 0$ for all $y \in \mathcal{C}$ since $s(y)$ is a SOS polynomial, and for all points of the safe set, i.e., $y \in \mathcal{C}$, we have $h(y) \geq 0$ as per the definition of the safe set characterized by a continuously differentiable function in Section 1. Hence the requirement that $u \in \widetilde{\mathcal{U}}(x)$ implies

$$\frac{\partial E(x,y)}{\partial x}(f(x) + g(x)u) + \alpha_1(E(x,y)) + \alpha_2(h(y)) \geq 0$$

for all $y \in \mathcal{C}$, i.e., $u \in \mathcal{U}(x)$ as defined in (7.1). In addition, since for all $x \in \mathcal{D}$, the constraint in $\widetilde{\mathcal{U}}(x)$ is convex in u, $\|u\|$ is convex in u, and k is continuous in x, using [73, Proposition 8], we conclude that the controller is continuous. From Theorem 10, $\mathcal{E}(x(t)) \subseteq \mathcal{C}$ for all

$t \geq 0$. ■

To implement the SOS controller in, e.g., SOSTOOLS [107], the degree of the SOS decision polynomial $s(y)$ in the constraint defining $\widetilde{\mathcal{U}}(x)$ is fixed a priori. In addition, the quadratic cost in (7.3) is recast in epigraph form to obtain an equivalent problem with linear cost and an additional semidefinite constraint via *Schur complement* [114]; in particular, the initial formulation (7.3) is equivalent to

$$u_{\text{SOS}}(x) = \underset{u \in \widetilde{\mathcal{U}}(x)}{\arg\min} \; \min_{\delta \in \Delta(u,x)} \delta \,,$$

with

$$\Delta(u,x) \quad = \quad \left\{ \delta \in \mathbb{R} \;\middle|\; \begin{bmatrix} I & u \\ u^T & \delta + 2k(x)^T u - k(x)^T k(x) \end{bmatrix} \geq 0 \right\}.$$

The above SOS approach allows us to adopt a tractable method to guarantee safety for the system. However, this approach has two drawbacks. First, it requires extent sets and safe sets to be defined by polynomial functions. Second, with increasing system dimensionality, SOS programs are known to become computationally difficult. Hence, we next propose a computationally efficient sampling based approach as an alternative that accommodates arbitrary extent functions and barrier functions.

7.3.2 A Sampling Based Approach to Set Invariance with Extent

In this subsection, we propose an alternative relaxation of (7.2) which retains the computational advantages of the original QP formulation for ZCBFs. The intuition is to enforce the constraint in (7.1) on the boundary of the extent set, but only for a finite number of sampled points. To obtain a finite number of sampled points, we discretize the boundary of the extent set $\partial\mathcal{E}(x)$. The following theorem guarantees safety by introducing a constraint on the control input for each sampled point.

Theorem 12. *Consider system* (1.1) *with initial state* $x(0)$*, an extent function* E*, and an Ec-CBF* h *with associated safe set* $\mathcal{C} = \{x \in \mathcal{D} \mid h(x) \geq 0\} \subset \mathcal{D}$*. Further assume that the domain* $\mathcal{D}$ *is bounded,* $\mathcal{E}(x)$ *is bounded for all* $x \in \mathcal{D}$*, let* $M > 0$ *be a bound on the magnitude of the control input, and for some* $\tau > 0$ *let* $\partial\mathcal{E}_\tau(x) \subset \partial\mathcal{E}(x)$ *be a finite set such that for all* $y \in \partial\mathcal{E}(x)$*, it holds that* $\min_{\widetilde{y}\in\partial\mathcal{E}_\tau(x)} \|\widetilde{y} - y\| \leq \tau/2$*. Additionally, let*

$$A \geq \sup_{y\in\mathcal{D}} \|\frac{\partial}{\partial y} h(y)\|, \tag{7.4}$$

$$B \geq \sup_{x,y\in\mathcal{D},\,|u|<M} \|\frac{\partial^2 E(x,y)}{\partial x \partial y}(f(x) + g(x)u)\|. \tag{7.5}$$

Consider the set

$$\widehat{\mathcal{U}}(x) = \Big\{ u \in \mathbb{R}^m \;\Big|\; \|u\| \leq M \text{ and } \frac{\partial E(x,y^*)}{\partial x}(f(x) + g(x)u) + \gamma \cdot h(y^*) \geq (B + \gamma A)\tau \text{ holds for all } y^* \in \partial\mathcal{E}_\tau(x) \Big\}$$

where $\gamma > 0$ *is a constant. Let* $k : \mathcal{D} \to \mathbb{R}$ *be a continuous nominal control input. If for all* $x \in \mathcal{D}$ *the set* $\widehat{\mathcal{U}}(x)$ *is non-empty and* $\mathcal{E}(x(0)) \subset \mathcal{C}$*, then the solution* $x(t)$ *to the system* (1.1) *with* $u = u_{sampled}(x)$*, where*

$$u_{sampled}(x) = \arg\min_{u\in\widehat{\mathcal{U}}(x)} \|u - k(x)\|_2^2, \tag{7.6}$$

is such that $\mathcal{E}(x(t)) \subseteq \mathcal{C}$ *for all* $t \geq 0$*. Moreover, the controller* $u_{sampled}(x)$ *is continuous with respect to* x *for all* $x \in \mathcal{D}$*.*

Proof. Introduce the open set

$$\mathcal{B} = \bigcup_{y^*\in\partial\mathcal{E}_\tau(x)} B^o_{\frac{3\tau}{4}}(y^*),$$

where $B^o_{\frac{3\tau}{4}}(y^*)$ denotes an open ball with radius $\frac{3\tau}{4}$ centered around y^*. Choose $\epsilon > 0$ such that for all x, $\bar{\mathcal{E}}(x) = \{y \in \mathcal{D} \mid |E(x,y)| \leq \epsilon\} \subseteq \mathcal{B}$. Such a choice of ϵ is possible due to part 3 of the definition of the extent function (Definition 11). Clearly $\partial\mathcal{E}_\tau(x) \subset \partial\mathcal{E}(x) \subset \bar{\mathcal{E}}(x)$. From the mean value theorem, for all $y \in \bar{\mathcal{E}}(x)$ and $y^* = \arg\min_{\bar{y}\in\partial\mathcal{E}_\tau(x)} \|\bar{y} - y\|$, it follows that

$$h(y^*) - h(y) \leq A\|y^* - y\|\,, \tag{7.7}$$

and

$$\left(\frac{\partial E(x,y^*)}{\partial x} - \frac{\partial E(x,y)}{\partial x}\right)(f(x) + g(x)u) \leq B\|y^* - y\| \tag{7.8}$$

whenever $|\, u \,| \leq M$. Now, observe that $\|y^* - y\| \leq 3\tau/4$. Multiplying (7.7) with $-\gamma$ and (7.8) with -1 and then adding the inequalities yields

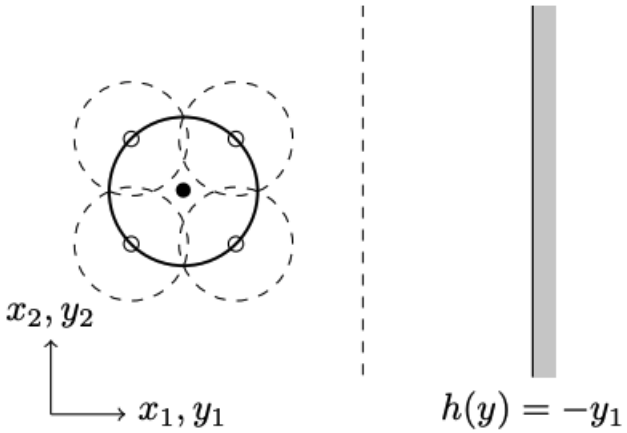
$$\begin{aligned}
&\frac{\partial E(x,y)}{\partial x}(f(x) + g(x)u) + \gamma \cdot h(y) \\
&\geq \frac{\partial E(x,y^*)}{\partial x}(f(x) + g(x)u) + \gamma \cdot h(y^*) - (B + \gamma A)3\tau/4 \\
&\geq (B + \gamma A)\tau/4
\end{aligned}$$

for any $u \in \widehat{\mathcal{U}}(x)$ where the last inequality follows from the definition of $\widehat{\mathcal{U}}(x)$.

Choosing $\alpha_1(s) = (B + \gamma A)\tau/(4\epsilon)s$ and $\alpha_2(s) = \gamma s$, for any $u \in \widehat{\mathcal{U}}(x)$, we have $\mathcal{M}(x,y,u) \geq 0$ for all y such that $|E(x,y)| \leq \epsilon$ where $\mathcal{M}$ is as in Definition 12, and therefore, in accordance with Remark 5, h is a Ec-CBF and $u_{\text{sampled}}(x) \in \mathcal{U}(x)$ for all $x \in \mathcal{D}$. Since for all $y^* \in \partial\mathcal{E}_\tau(x)$, in the definition of $\widehat{\mathcal{U}}(x)$, the constraint

$$\frac{\partial E(x,y^*)}{\partial x}(f(x) + g(x)u) + \gamma \cdot h(y^*) \geq (B + \gamma A)\tau$$

is convex in u, $|\, u \,|$ is convex in u and k is continuous in x, using [73, Proposition 8], we conclude that the controller $u_{\text{sampled}}(x)$ is continuous. Thus, from Theorem 10, the extent set $\mathcal{E}(x)$ is contained within the safe set for all $t \geq 0$ and for all $x \in \mathcal{D}$ such that



Example 1.

Figure 7.2: A motivating example, where a differential drive robot with a system volume covered by an extent set must stay within the ellipsoidal safe set when a nominal unsafe controller tries to drive it outside. This chapter proposes controller frameworks to guarantee safety of the system including its volume under such situations. The image is from a Robotarium [67] implementation of the framework introduced in this chapter, which we also describe in Section 8.5.

$E(x(0)) \subseteq \mathcal{C}$; that is, $\mathcal{E}(x(t)) \subseteq \mathcal{C}$ for all $t \geq 0$. ■

The constants (7.4) and (7.5) are interpreted as Lipschitz constants for functions appearing in the definition of Ec-CBF. Whenever the domain $\mathcal{D}$ is bounded, as is usually the case in practice, such constants will exist. Moreover, the constant τ must be chosen small enough so that $\widehat{\mathcal{U}}(x)$ is nonempty and, as we show in the following example, choosing τ to be large can result in unwanted conservatism.

Consider the system $\dot{x} = u$*, where* $x = \begin{bmatrix} x_1 & x_2 \end{bmatrix}^T \in \mathbb{R}^2$ *is the system state and* $u \in \mathbb{R}^2$ *is a bounded control input such that* $||u_2|| \leq M$ *where* $M = 1$*. We take* $E(x, y) = (x_1 - y_1)^2 + (x_2 - y_2)^2 - 1$ *and encode the safety constraint with the safe function* $h(y) = -y_1$*. This problem setting is depicted in Fig. 7.2. We take Lipschitz constants* $A = 1$ *and* $B = 2$ *satisfying* (7.4) *and* (7.5)*. Observe that for this specific choice of extent function, the constants can be determined although the domain is unbounded. In practice, systems operate in a compact domain, and hence, the constants* A *and* B *exist and can be computed.*

To demonstrate the conservatism with few sampled points, we consider a sampling

based controller which uses four samples; that is, we take $\partial\mathcal{E}_\tau(x) = \{(x_1 \pm 1/\sqrt{2}, x_2 \pm 1/\sqrt{2}), (x_1 \pm 1/\sqrt{2}, x_2 \mp 1/\sqrt{2})\}$*, such that* $\tau = 2\sqrt{2-\sqrt{2}}$*. The controller* (7.6) *then has four constraints,*

$$\sqrt{2}(-u_1 \pm u_2) - \gamma\left(x_1 - \sqrt{2}\right) \geq (B + \gamma A)\tau\,, \tag{7.9}$$

$$\sqrt{2}(u_1 \pm u_2) - \gamma\left(x_1 + \sqrt{2}\right) \geq (B + \gamma A)\tau\,. \tag{7.10}$$

In the instance that $x_1 = -(\frac{B}{\gamma} + A)\tau$*, the only feasible solution is* $(u_1, u_2) = (-\gamma, 0)$ *with* $\gamma \leq M$*, which will effectively steer the system away from the barrier. Observe that* γ *plays a key role in the behavior of the system. A higher value of* γ *will allow for the system to get closer to the barrier, but once the system is close to the boundary, a more aggressive control action, i.e.,* $u_1 = \gamma$ *is applied.*

Intuitively, the sampling-based technique essentially covers the boundary of the extent set with balls around the discretized points, and ensures that the balls do not ever cross over into the unsafe set, as is visualized in Fig 7.2.

7.4 Experimental Results

In this section, we present a case study[1] of the proposed framework implemented in the Robotarium testbed [67] on a differential drive robot with dynamics

$$\dot{x}_1 = v \cdot \cos(\phi)\,, \quad \dot{x}_2 = v \cdot \sin(\phi)\,, \quad \dot{\phi} = \omega\,,$$

where $x_1 \in \mathbb{R}$, $x_2 \in \mathbb{R}$ are the position coordinates of the robot, $\phi \in [-\pi, \pi)$ is the orientation, and $v \in \mathbb{R}$ and $\omega \in \mathbb{R}$ are the linear velocity input and angular velocity input. Define $\widehat{x} = \begin{bmatrix} x_1 & x_2 \end{bmatrix}^T$, and $x = \begin{bmatrix} x_1 & x_2 & \phi \end{bmatrix}^T$. The Robotarium domain $\mathcal{D}$ is defined by $x_1 \in [-1.6, 1.6]$, $x_2 \in [-1, 1]$ and $\phi \in [-\pi, \pi)$. We consider an ellipsoidal safe set $\mathcal{C} = \{\widehat{x} \in \mathbb{R}^2 \mid$

[1] Code – `https://github.com/gtfactslab/ExtentCBF`

Table 7.1: Average computation time for the controllers in the case study

No. of Samples	**Average Run time (in milliseconds)**
50	5.5
100	7.2
500	19.0
1000	31.9
2000	38.8
SOS-based controller	1806.5

$h(\widehat{x}) \geq 0\}$ where $h(x) = 1 - \widehat{x}^T P_{\text{safe}} \widehat{x}$ with $P_{\text{safe}} = \text{diag}(1^{-2}, 0.8^{-2}) \in \mathbb{R}^{2\times 2}$, and the extent set as a fourth-order superellipse described by the extent function

$$E(x,y) = (7.5)^4(\Delta_1 \cos(\phi) + \Delta_2 \sin(\phi))^4 + (10)^4(-\Delta_1 \sin(\phi) + \Delta_2 \cos(\phi))^4 - 1\,, \tag{7.11}$$

where $\Delta_i = (x_i - y_i)$ for $i = 1, 2$.

We take the nominal control as $u_{\text{nom}} = \begin{bmatrix} 1 & 0.4 \end{bmatrix}^T$, with a simulation horizon of 1000 iterations. We compare the trajectories generated from the SOS-based approach and the sampling-based approach in Fig 7.4. For this particular choice of nominal control input, the SOS-based controller makes slower progress along the desired trajectory; this is because, once the robot gets closer to the safe set boundary, the safe input velocities becomes small. For the sampling based controller, we take $\gamma = 0.4$ and considered 50, 100, 500, 1000 and 2000 samples. The average time required to compute the control law for the controllers is shown in Table 7.1. In fact, even with 5000 sampled boundary points, the average computation time of the QP (7.6) was 56.3 milliseconds, suggesting that the sampling-based method is well-suited for real-time implementation.

In Fig. 7.3, we plot the minimum value of the Ec-CBF evaluated over all sampled points for cases in Table 7.1. Thus, to generate this plot, we considered 50, 100, 500, and 2000 sampled points on the boundary. Observe that all the values are strictly positive, thus implying that none of the sampled points have violated the safe set.

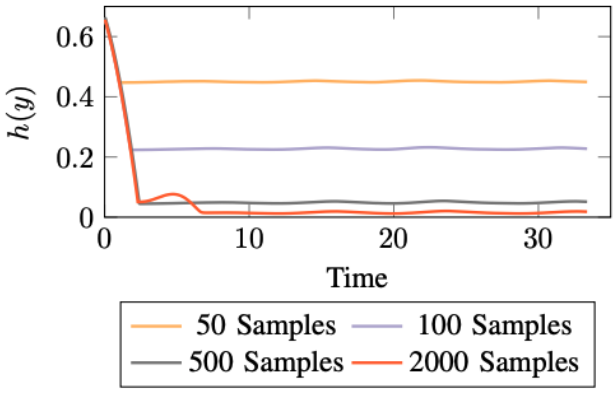


Figure 7.3: Minimum value of the Ec-CBF $h(y)$ evaluated for 50, 100, 500, and 2000 sampled points (i.e. for all $y \in \partial\mathcal{E}_\tau(x)$) on the boundary of the extent set. Observe that all values for each case are strictly positive, thus implying that none of the sampled points cross into the unsafe region.

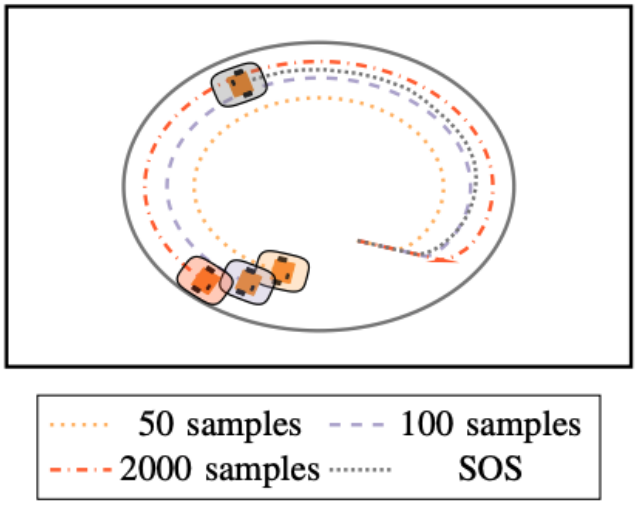


Figure 7.4: A collection of trajectories for the sampling-based controller, generated with different number of samples. Observe that more number of samples reduces the conservatism, which allows the robot to approach the boundary more closely. The SOS-based trajectory is included for reference.

The nature of the robot trajectory is influenced by the number of sampled points considered. In Fig. 7.4, we show how the trajectories for the robot differ with 50, 100 and 2000 sampled points. As expected and per our discussion in Example 1, we observe that there is larger conservatism when lower number of samples are considered. The robot does not venture close to the boundary of the safe set. However, with large number of samples, we observer that the robot gets very close to the boundary. From Table 7.1, we note that there is not a significant increase in computation times for large number of sampled points, and hence, one can safely use the sampling-based controller to avoid conservatism.

From the above discussion, we observe that using the sampling-based controller results in a computationally efficient controller which guarantees safety. To showcase this, we implemented[2] the sampling-based technique on the Robotarium testbed as in Fig 7.1. In particular, we considered 500 samples for the implementation. As can be seen, the trajectory of the robot is such that safety of the system volume is guaranteed over the period of the entire experiment. The average computation time for the QP was 5.1 milliseconds.

7.5 Concluding Remarks

This chapter presents a barrier function based method for ensuring the safety of a control-affine dynamical system that incorporates its physical volume, i.e., its extent. The first resulting controller design relies on a sum-of-squares based optimization program. Since sum-of-squares programs can be computationally difficult, a sampling based method is also presented. The proposed sampling based controller is shown to retain the guarantee on safety of the system. Simulation and robotic implementation results are also provided.

[2]Experiment Video- `https://youtu.be/LUp-vZ7-eJg`

Part IV

Machine Learning and Barrier Functions

CHAPTER 8

LEARNING CONTROL BARRIER FUNCTIONS FOR PATH PLANNING TASKS

In this section, a key assumption prevalent in barrier function-based approaches is discussed-complete knowledge of the barrier function. This is a fairly restrictive assumption, especially when such frameworks are to be used in the context of autonomous systems operating in unknown environments. To that end, this chapter will present a framework that synthesizes barrier functions using sensor inputs from the environment.

Introduction

Autonomous vehicles [115], industrial robots, and multi-robot systems [116] deployed in uncertain domains are often tasked to respect safety-critical constraints while advancing a given task [117]. When operating in unknown and dynamic environments with insufficient advanced information regarding the workspace, controllers which translate sensory information from the environment into *safe* control actions are of paramount importance. Control barrier functions (CBFs) are level-set functions used to provide formal safety guarantees for controlled dynamical systems. Given a possibly unsafe nominal control policy, barrier function based quadratic programs (QPs) generate a *safe* control action at each time instant. The control actions are minimally invasive in the sense that the nominal control policy is modified only when it will result in violation of a safety constraint. Barrier function based real-time controllers in robotics support collision avoidance for multi-robot motion [118], task allocation for robotic swarms [97], and motion planning [32].

A key assumption commonly imposed is that the robotic system has complete knowledge of the unsafe state space regions. Leveraging the knowledge translates to formal safety guarantees arising from its translation to CBFs. In practice, this assumption need not hold and limits more widespread application of barrier functions. As a motivating example, con-

sider an autonomous robot operating in an environment for which it has no knowledge of the obstacle boundaries. If these boundaries are to be as level-sets of smooth functions, the process of finding closed-form barrier functions for these obstacles is not straightforward. Without the functions, one cannot leverage the safety guarantees that CBFs provide. Thus, this chapter describes a support vector machine (SVM) approach to CBF synthesis from sensor measurements. In particular, sensory information obtained from the environment defines the set of safe and unsafe samples and is used for training the SVM classifier.

Learning algorithms, or data-driven synthesis methods, for ensuring safety have been explored in several contexts. The most prevalent has been to establish stable state space regions meeting safety specifications by identifying a control Lyapunov function (CLFs) compatible with given CBFs. Techniques for doing so include sum-of-squares (SOS) methods [119] and neural network designs [120], with the aim of identifying the largest possible stable region within the safe region. When attempting to learn baseline control policies for a given task, reinforcement learning methods cannot guarantee safety as the exploration process demands executing unsafe control inputs. Employing pre-specified barrier functions during the action policy exploration and keeping track of the safety interventions to influence the explored policies, can provide the necessary safety guarantees [121]. Investigations more closely aligned with barrier function synthesis using tools from machine learning include the use of kernel machines [122] to synthesize occupancy map functions for navigation and planning purposes [123, 124]. Occupancy map level-sets can distinguish safe and unsafe regions. This potential use was further explored in the context of perceptron algorithms, where the resulting classifier function provided a mechanism to synthesize non-colliding trajectories through space [125]. Emphasis was on improving the run-time of global mapping relative to existing kernel machine methods. Our aim is to explore how these machine learning constructs can be used to synthesize CBFs in a manner that the learned function provides the necessary safety guarantees.

The contributions of this chapter are as follows: First, we present a SVM approach

for the synthesis of a barrier function from a training dataset consisting of *safe* and *unsafe* samples obtained from sensor measurements. We describe offline and online training methods. Second, a formal guarantee on correct classification of unsafe regions is provided for both the methods. We show that in the offline method, the system is rendered safe for an under-approximated (conservative) safe set. A similar guarantee holds locally in the online approach. The proposed framework is implemented in a ROS-based simulator with a LiDAR equipped omnidirectional robot. Evaluation metrics for the trajectories generated by the proposed CBF synthesis framework quantify how well they match the ideal case where the CBF is known. To the best of our knowledge, this is the first chapter addressing the problem of CBF synthesis from sensed environmental data.

8.1 Problem Setup

Consider an affine in control, robotic system as in (1.1) evolving in $\mathcal{D} \subset \mathbb{R}^2$ and equipped with LiDAR sensors that provide depth information. By virtue of the depth measurement vector $z_t \in \mathbb{R}^N_{>0}$ at time t, where N is the total number of samples, the robot can detect unsafe state space regions. Regarding the LiDAR sensor, denote by θ_{res} the angular resolution (increment angle) of the measurements. This is the angle between two consecutive light rays emitted from the sensor. We make the following assumption in order to account for spatial variations in the nature of the workspace: assume that the resolution of the LiDAR sensor is high enough to capture the spatial profile of the environment from a given offset distance, i.e., the LiDAR has a sufficiently small increment angle θ_{res}. Sensors such as the ones from Velodyne [126] with increment angles as small as 0.08° are capable of satisfying the above assumption. Hence, it is reasonable to assume such sensor resolution capabilities.

Let $k \in C(\mathcal{D}; \mathbb{R}^m)$ be a user-defined nominal feedback control policy to be followed by the robot. Examples of such policies include proportional (go-to-goal) control or MPC based policies [127]. The state space is assumed to contain unknown unsafe regions. That is, there exist p unsafe sets in the state space defined as $\mathcal{C}_i = \{x \in \mathcal{D} \mid h_i(x) \leq 0\, h_i \in$

$C(\mathcal{D};\mathbb{R})\}$ for all $i \in \{1,2,\ldots,p\}$, such that h_i are unknown ZCBFs. The safe region is $\mathcal{D}\setminus \cup_{i=1}^{p}\mathcal{C}_i$.

Since there is no *a priori* knowledge of the unsafe sets, data obtained from the LiDAR sensor must be used to synthesize the unknown barrier functions $h_i : \mathcal{D} \to \mathbb{R}$, $i \in \{1,2,\ldots,p\}$, to render the system safe while minimally deviating from the nominal feedback policy k. In conjunction with the robot's state, the measurements obtained from the on-board depth sensors provide the location of points on the boundary of the unsafe sets, and hence are points $x \in \mathcal{D}$ for which $h(x) = 0$. To learn the unsafe regions and follow the nominal policy safely, a framework for the synthesis of barrier functions is required with guarantees on safety of the system, as formalized by the problem statement:

Problem Statement 6. *Consider the affine in control, robotic system in* (1.1) *and the unsafe sets $\mathcal{C}_i \subset \mathcal{D}$, $i \in \{1,2,\ldots,p\}$. Given the nominal feedback control policy $k : \mathcal{D} \to \mathbb{R}$ and LiDAR measurements z_t obtained at any time instant $t \geq 0$, formulate a barrier function synthesis framework which either*

1. *Learns the unsafe region $\bigcup_{i=1}^{p}\mathcal{C}_i$ offline given a dataset of safe and unsafe samples from the domain, or*

2. *Learns the unsafe region online using instantaneous measurements z_t, as the system traverses the domain.*

8.2 Control Barrier Functions Synthesis Framework

This section describes the methodology for obtaining the training dataset, the control barrier function synthesis framework, and two QP based approaches which utilize the synthesized barrier function to guarantee safety.

Support Vector Machines

The learning approach to be used for barrier function specification via-a-vis the unsafe regions will be support vector machine (SVMs), namely kernel SVMs [122]. Suppose a dataset $\mathcal{T} = \{(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)\}$ is provided where $x_i \in \mathbb{R}^n$ is an n dimensional vector and $y \in \mathcal{Y} = \{-1, 1\}$ is a label associated with the vector x_i for all $i \in \{1, 2, \ldots, N\}$. Using the dataset, the linear SVM algorithm determines an affine decision boundary function $\widehat{f}(w^T x + b)$, where $x \in \mathbb{R}^n$ is a training sample, $w \in \mathbb{R}^n$ are coefficients and $b \in \mathbb{R}$ is a bias term, which translates the sample x into a corresponding label $y \in \mathcal{Y}$ that belongs to one of the two classes i.e. $+1$ or -1. When the data is not separable by a hyperplane in the native space, a non-linear mapping transforming the data into a higher dimensional space with better separability properties may be used. This chapter makes use of such a mapping, via a kernel function, to facilitate separation of unsafe obstacle regions from safe regions.

Since the domain $\mathcal{D}$ consists of states which are either safe or unsafe, their separation can be cast as a binary SVM classification problem. However, it is imperative that unsafe states be classified as unsafe, whereas all the safe states need not strictly be classified as safe. To that end, we consider the non-linear, biased-penalty SVM optimization problem [128]:

$$\begin{aligned} \underset{w}{\text{minimize}} \quad & \frac{1}{2}\|w\|_2^2 + C^+ \sum_{i|y_i=+1}^{N} \xi_i + C^- \sum_{j|y_j=-1}^{N} \xi_j \\ \text{s.t} \quad & y_i \cdot (w\phi(x_i) + b) \geq 1 - \xi_i \\ & \xi_i \geq 0, \text{ for all } i \in \{1, 2, \ldots, N\}\ , \end{aligned} \tag{8.1}$$

where $C^+, C^- > 0$ are constants penalizing misclassification of the positive and negative samples, and $\phi : \mathbb{R}^n \to \mathbb{R}^d$ is a non-linear mapping into a higher dimensional space. In practice, the dual of the above optimization problem is solved by using a kernel function

k_ϕ to bypass the need to explicitly define ϕ [122]. We use the Gaussian kernel,

$$k_\phi(x_i, x_j) = \exp\left(- \frac{\|x_i - x_j\|^2}{\sigma^2} \right), \tag{8.2}$$

where $\sigma > 0$ is the bandwidth of the kernel (and is a hyper parameter).

Observe that in (8.1) there are two separate costs for the positive and negative classes. Unequal costs permit a greater bias towards correctly classifying one class over the other. In particular, having $C^- = \infty$ and $\infty > C^+ > 1$ induces a hard margin classification for the unsafe states and allows for some misclassification for the safe states. This outcome is captured by the so called *cost matrix* (M) of the form

True \ **Estimated**	Safe	Unsafe
Safe	0	C^+
Unsafe	C^-	0

Each entry $[M]_{ij}$ of the matrix represents the cost of classifying a sample as label j when it truly belongs to label i. The penalty for classifying a truly safe (or unsafe) state as safe (or unsafe) is zero. It is undesirable to classify a truly unsafe state as safe, motivating a high penalty for C^-. Since safe states being classified as unsafe do not compromise safety, the penalty C^+ may be smaller. The optimization problem (8.1) provides compliance (in favor of safety) to measurement errors and noise in the sensor data which can affect the generated decision boundary. The mixed hard/soft margin classification is what supports the theoretical safety guarantees of the system as discussed in the following subsections.

8.2.1 Training Dataset Generation

This section details the training data generation process suited to binary SVM classification per (8.1). A pictorial example for obtaining the training data from a LiDAR sensor at a particular time instant is shown in Fig. 8.1. Below we provide a detailed explanation for generating the dataset.

Generating meaningful data for the kernel SVM from the LiDAR sensor requires converting the scalar variables into world Cartesian coordinates by means of a laser scan transform $g : \mathbb{R} \times \mathcal{D} \to \mathbb{R}^2$, whose main input is the laser scan measurements in polar coordinates and the current robot state (for mapping from the robot frame to the world frame). Assume that if the sensor detects an unsafe region, then the output from the sensor is a finite depth reading, else it is infinite. In particular, given a measurement vector $z_t = \begin{bmatrix} z_t^1 & z_t^2 & \dots & z_t^N \end{bmatrix}^T \in \mathbb{R}^N$ at time t with N samples, define $\mathcal{F} \subset \mathcal{I} = \{1, \dots, N\}$ to be the index set of the finite scan measurements. Define $\mathcal{O}^- = \bigcup_{i \in \mathcal{F}} \{g(z_t^i; x_t)\}$ as the set of unsafe samples. $\mathcal{O}^-$ represents points on the boundary of the unsafe set detected by the sensor which is used to populate a dataset of negative labeled samples $\mathcal{T}^- = \bigcup_{i \in \mathcal{F}} \{(g(z_t^i; x_t), -1)\}$. To obtain the positive samples from the environment, each $g(z_t^i; x_t) \in \mathcal{O}^-$ is projected radially backwards along the line segment joining the state of the robot $x(t)$ and the point $g(z_t^i; x_t)$, by a finite distance $d \in \mathbb{R}_{>0}$. Define

$$\widehat{z}_t^i = g(z_t^i - d; x_t) \in \mathbb{R}^2 \tag{8.3}$$

for all $i \in \{1, 2, \dots, N\}$ where $d > 0$ is the finite offset distance. Define the set of positive samples as $\mathcal{O}^+ = \bigcup_{i \in \mathcal{F}} \{\widehat{z}_t^i\}$, with the dataset for positive labeled samples constructed as $\mathcal{T}^+ = \bigcup_{i \in \mathcal{F}} \{(\widehat{z}_t^i, +1)\}$. Collecting the set of positive and negative labeled samples generates the training dataset $\mathcal{T} = \mathcal{T}^+ \cup \mathcal{T}^-$. The training dataset $\mathcal{T}$ contains all unsafe samples and corresponding safe samples for training the SVM classifier. The procedure is summarized in Algorithm 7.

8.2.2 Barrier Function Synthesis with Kernel-SVMs

To improve the ability to capture unsafe region boundaries, the point data is transformed by a fixed set of Gaussian kernels of the form (8.2) using a sparse set of grid points over the domain $\mathcal{D}$. This provides a first kernel machine layer that behaves like an approximate

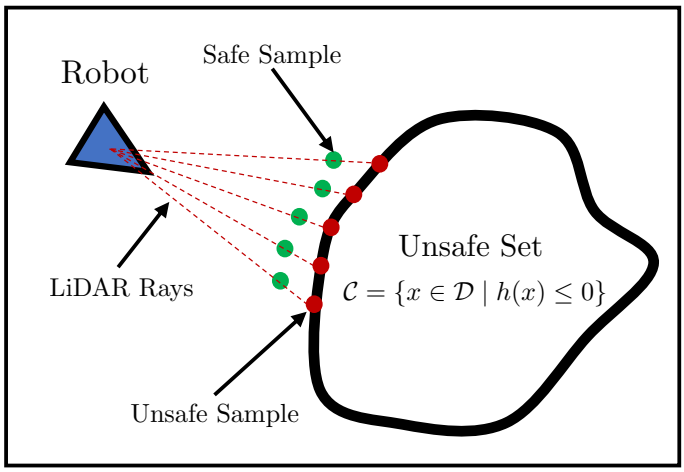


Figure 8.1: A particular instantiation of a training dataset obtained from measurements from a LiDAR sensor for a given unsafe set. The red points indicate unsafe samples which represent the boundary of the unsafe set whereas the green points indicate the safe samples obtained by the transformation dictated by equation (8.3). The red dashed lines indicate the LiDAR rays emanating from the sensor onboard the robot.

Algorithm 7 Training Dataset Generator

Input : Laser Scan Measurement z_t **and Robot State** x_t
Output: Training Dataset $\mathcal{T}$

1: Identify $\mathcal{F} \subset \{1, \ldots, N\}$
2: $\mathcal{T}^- = \bigcup_{i \in \mathcal{F}} \{(g(z_t^i; x_t), -1)\}$
3: $\widehat{z}_t^i = g(z_t^i - d; x_t)$, $\forall i \in \mathcal{F}$
4: $\mathcal{T}^+ = \bigcup_{i \in \mathcal{F}} \{(\widehat{z}_t^i, +1)\}$
5: $\mathcal{T} \leftarrow \mathcal{T}^- \cup \mathcal{T}^+$
6: Return $\mathcal{T}$

Hilbert space occupancy map [123] and roughly captures the different safe and unsafe regions of the state space. Passing the vector output of this Hilbert space to the kernel SVM generates a second layer that can refine the boundary to better separate the safe and unsafe regions. The solution to the hard/soft margin kernel SVM in (8.1) defines the parameters for a non-linear decision boundary separating the training data (the output layer of the full classification network). Evaluating the two-layer classifier model for $x \in \mathcal{D}$ outputs a posterior probability describing the likelihood that the sample $x \in \mathcal{D}$ belongs to a particular class i.e., safe or unsafe. The posterior probabilities obtained from the model are then converted into margin scores which define a signed level-set function and provide the barrier function we seek. The barrier function approximator is thus a two hidden layer Gaussian kernel neural network. This entire procedure is summarized in Algorithm 8. By virtue of the methodology used to generate the training data in Algorithm 7, and the biased-penalty hard margin SVM optimization problem (8.1), the synthesized barrier function correctly classifies the unsafe samples. This is formalized in Proposition 6.

Proposition 6. *Given a training dataset $\mathcal{T}$ obtained from Algorithm 7, if Algorithm 8 is used to synthesize the barrier function $\widehat{h}$, then the unsafe samples $\mathcal{O}^-$ are such that $\widehat{h}(x) < 0$ for all $x \in \mathcal{O}^-$.*

Proof. By the method presented in Algorithm 7 to generate the training dataset $\mathcal{T}$, we have that the set $\mathcal{O}^-$ consists of points on the boundary of the unsafe set. From the kernel-SVM approach used in Algorithm 8, a function $\widehat{h}$ is generated which classifies the *safe* and *unsafe* samples. Since the optimization problem (8.1) is a hard margin SVM for the unsafe samples and RBF kernels have universal function approximation capabilities (Theorem 2, [129]), we can guarantee that $\widehat{h}(x) < 0$ for all $x \in \mathcal{O}^-$ and thus the proposition follows. ■

Algorithm 8 Kernel-SVM based Barrier Function Synthesis

Input : Training Dataset $\mathcal{T}$
Output: Estimated Barrier Function $\widehat{h}$

1: $\mathcal{T}_{HS} \leftarrow$ Map samples in $\mathcal{T}$ to Hilbert space
2: $Cl \leftarrow$ Train kernel SVM classifier (8.1) using $\mathcal{T}_{HS}$
3: $\widehat{h} \leftarrow$ Recover signed distance function from Cl and the first Gaussian kernel layer mapping
4: Return $\widehat{h}$

8.3 Offline Barrier Function Synthesis & Control

Here, we discuss the *offline* approach to CBF synthesis using Algorithm 7 and Algorithm 8. Per the problem setup in Section 8.1, we consider the workspace consisting of p unsafe regions characterized by ZCBFs h_i, $i \in \{1, 2, \ldots, p\}$. We assume that there exists an *oracle* which provides a set of unsafe samples corresponding to the boundary of each unsafe sets $i \in \{1, 2, \ldots, p\}$ in the state space by means of a LiDAR sensor which are dense enough to cover the true boundary of the obstacles. For example, this oracle can be a "mapping" robot that navigates the domain and gathers data about the safe and unsafe regions.

Once Algorithm 7 generates the requisite training data using the *oracle*, executing Algorithm 8 leads to a ZBF estimate. Note that a single ZCBF $\widehat{h} \in C^1(\mathcal{D}; \mathbb{R})$, whose zero level-set captures the boundaries between safe and unsafe regions, is obtained as opposed to p different ZCBFs characterizing the unsafe sets. With the synthesized barrier function $\widehat{h}$, we then implement a QP controller with (1.3) as the constraint. Capturing all the unsafe sets with a single function means that the QP involves only one constraint which reduces the computational complexity involved in computing the control input. The QP is solved, and the control is applied, until the system completes the specified task associated to the nominal controller. The entire offline barrier function synthesis and control methodology is formalized in Algorithm 9. In the algorithm, the initial loop from $t = 0$ to $t = T$ where $T < \infty$, indicates the time period when the training data is gathered for generating the barrier function.

Recall that the increment angle of the LiDAR sensor is given by θ_{res}. Intuitively, as

Algorithm 9 Offline SVM-based QP controller

Input : Nominal controller k

$\mathcal{T} \leftarrow \varnothing$
for $t \in [0, T]$ **do**
  $z_t \leftarrow$ `LaserScanMeasurement`
  $\mathcal{T}_t \leftarrow$ `TrainingDataGenerator`(z_t, x_t)
  $\mathcal{T} \leftarrow \mathcal{T} \cup \mathcal{T}_t$
end for
$\hat{h} \leftarrow$ `BarrierEstimator` $(\mathcal{T})$
while `Goal is not reached` **do**
  Solve the QP:

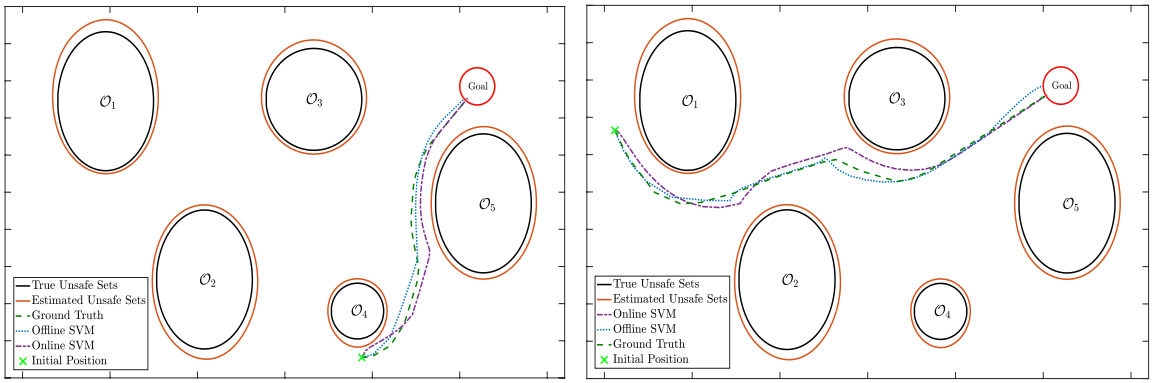
$$u^*(x) = \underset{u \in \mathbb{R}^m}{\operatorname{argmin}} \quad \|u - k(x)\|_2^2$$
$$\text{s.t} \quad L_f\widehat{h}(x) + L_g\widehat{h}(x)u(x) \geq -\alpha(\widehat{h}(x))$$

  $u \leftarrow u^*(x)$
  Solve (1.1), update state $x(t)$
end while

$\theta_{\text{res}} \to 0$, the LiDAR sensor captures the true nature of the boundary of the unsafe region. Hence, using Proposition 6, we can guarantee that Algorithm 8 synthesizes a barrier function whose level-sets are over-approximations of the true unsafe regions. That is, denote $\widehat{S} = \{x \in \mathcal{D} \mid \widehat{h}(x) \leq 0\}$ where $\widehat{S} : \mathcal{D} \to \mathbb{R}$ as the unsafe region estimated by Algorithm 8. Then, we have that $\mathcal{S} \subset \widehat{\mathcal{S}}$, where $\mathcal{S} = \bigcup_{i=1}^{p} \{x \in \mathcal{D} \mid h_i(x) \leq 0\}$ is the true unsafe region characterized by the unknown barrier functions h_i for all $i \in \{1, 2, \ldots, p\}$. In practice, this statement holds true for high resolution LiDAR sensors. The degree of over-approximation depends on a number of factors which include the distance $d \in \mathbb{R}_{>0}$ with which the positive samples are generated in Algorithm 7. Next, we provide a formal guarantee that Algorithm 9 guarantees safety of the system.

Theorem 13. *Suppose $\mathcal{S} \subset \widehat{S}$ and the controller from Algorithm 3 is used. Then given any $x(0) \in \widehat{\mathcal{S}}^c$ where $\widehat{\mathcal{S}}^c = \{x \in \mathcal{D} \mid \widehat{h}(x) \geq 0\}$, the robot trajectory is such that $x(t) \in \widehat{\mathcal{S}}^c$ for all $t \geq 0$.*

Proof. From Algorithm 9, the QP enforces the barrier function constraint (1.3) with $\widehat{h}$ as



(a) (b)

Figure 8.2: Trajectories generated for the robot in a five obstacle scenario. The robot must reach a goal region (red circle) which is known a priori. Three different trajectories are shown- the ground truth trajectory (dashed green), the offline kernel-SVM based controller trajectory (dotted blue), and the online kernel-SVM based controller trajectory (dash-dotted purple). For the initial condition on the left, the trajectories show high correlation values ($R_{\text{offline}} = 0.9992$, $R_{\text{online}} = 0.9777$, $R_{\text{offline-online}} = 0.9734$) and small Fréchet distance values ($F_{\text{offline}} = 0.0469$, $F_{\text{online}} = 0.0822$, $F_{\text{offline-online}} = 0.0853$) which indicate that the trajectories are highly similar to the ground truth trajectory. For the figure on the right, the trajectories once again show high correlation values ($R_{\text{offline}} = 0.9627$, $R_{\text{online}} = 0.8085$, $R_{\text{offline-online}} = 0.8946$) and small Fréchet distance values ($F_{\text{offline}} = 0.0665$, $F_{\text{online}} = 0.0840$, $F_{\text{offline-online}} = 0.1334$). Observe that estimated unsafe set is an over-approximation of the true unsafe sets, and hence Algorithm 9 guarantees collision free trajectories in the offline case, as per Theorem 13.

the ZCBF. Since the cost function of the QP is quasi-convex in u, the constraints are quasi-convex in u and the nominal policy k is continuous, from Proposition 8 in [73] we have that the generated control u is continuous. Hence from Theorem 1 and by assumption $\mathcal{S} \subset \widehat{\mathcal{S}}$, we have that the set $\widehat{\mathcal{S}}^c = \{x \in \mathcal{D} \mid \widehat{h}(x) \geq 0\}$ is rendered forward invariant. That is, we have that $x(t) \in \widehat{\mathcal{S}}^c$ for all $t \geq 0$. ■

8.4 Online Barrier Function Synthesis & Control

When access to the full set of unsafe samples from the environment is not available, a real-time barrier function synthesis method is preferable. Here, we describe an online approach to synthesizing barrier functions, based on Algorithm 10. For online ZCBF synthesis, the set of unsafe samples covering the boundary of all the unsafe regions is not known a priori.

Algorithm 10 Online SVM-based QP controller

Input : Aggregate Flag δ, Nominal controller k

$\mathcal{T} \leftarrow \varnothing$
while `Goal is not reached` **do**
  $z_t \leftarrow$ `LaserScanMeasurement`
  $\mathcal{T}_t \leftarrow$ `TrainingDataGenerator`(z_t)
  if $\delta = 1$ **then**
    $\mathcal{T} \leftarrow \mathcal{T} \cup \mathcal{T}_t$
  else
    $\mathcal{T} \leftarrow \mathcal{T}_t$
  end if
  $\hat{h} \leftarrow$ `BarrierEstimator` $(\mathcal{T})$
  Solve the QP:

$$u^*(x) = \underset{u \in \mathbb{R}^m}{\text{argmin}} \quad \|u - k(x)\|_2^2$$
$$\text{s.t} \quad L_f\widehat{h}(x) + L_g\widehat{h}(x)u(x) \geq -\alpha(\widehat{h}(x))$$

  $u \leftarrow u^*(x)$
  Solve (1.1), update state $x(t)$
end while

Hence, at time $t = 0$, the system is initialized with no information regarding the state space, except the nominal feedback control policy. At each time instant t, the system obtains the depth measurement z_t and generates the training dataset $\mathcal{T}$ via Algorithm 7. Then, Algorithm 8 synthesizes a local barrier function. Implementing the QP controller generates the control input at time instant t. In the next time instant, the same procedure repeats and a new barrier function is synthesized based on the updated sensor measurements.

Two variations of the online barrier function synthesis method can be implemented. In the first method, the depth sensor data for all previous time instances is deleted, and the QP is solved with only the immediately sensed measurements. The barrier function approximates the true safe region only locally i.e., in a neighborhood around the state x_t of the robot. In the second method, samples from the previous time instant are aggregated with the samples from the current time instant, with Algorithm 8 implemented with the incremented set. The two cases synthesize different barrier function at each time instant. For the data-aggregation case, the estimate of the barrier improves as the number of sam-

ples characterizing the unsafe regions increases. Advantages and drawbacks exist for both approaches. In the data aggregation case, one needs to continuously update the dataset with new measurements and this exhaustive data collection process can become computationally expensive unless one resorts to efficient ways to store data [130]. For the non data aggregation case, computation is faster but the estimate of the barrier function does not improve iteratively as the robot traverses the domain. Both procedures are described in Algorithm 10.

Define the sensing range of the sensor as $\mathcal{B}_r(x) = \{\overline{x} \in \mathcal{D} \mid \|x - \overline{x}\| \leq r\}$, where $r \in \mathbb{R}_{>0}$ is the sensing range of the robot. Similar to the discussion in the previous subsection, it can be guaranteed that if $\theta_{\text{res}} \to 0$, then locally, Algorithm 8 synthesizes a barrier function whose level-set over approximates the true unsafe region. That is, denote $\widehat{\mathcal{S}}_r(x) = \{x \in \mathcal{B}_r(x) \mid \widehat{h}(x) \leq 0\}$ where $\widehat{h} : \mathcal{D} \to \mathbb{R}$ is the estimated ZCBF from Algorithm 8. Then, as $\theta_{\text{res}} \to 0$, we have that $\mathcal{S}_r(x) \subset \widehat{\mathcal{S}}_r(x)$ for all $x \in \mathcal{D}$ locally within the ball $\mathcal{B}_r(x)$, where $\mathcal{S}_r(x) = \bigcup_{i=1}^{p} \{\overline{x} \in \mathcal{B}_r(x) \mid h_i(\overline{x}) \leq 0\}$ is the true unsafe region. In the online case, a statement similar to Theorem 13 cannot be made since the robot does not have access to the full set of samples that characterize the entire boundary of the unsafe set and hence, there is no guarantee that globally in the domain the generated level-sets are over-approximations of the true unsafe regions. However, since the robot dynamics are locally Lipschitz continuous, safety holds locally as seen in Fig 8.2.

8.5 Implementation Results

This section describes and discusses simulation results from a path planning perspective conducted on the "Simple Two Dimensional Robot (STDR) simulator"[1] which allows one to generate different maps and simulate synthetic LiDAR data. Two environments were created for use in STDR. The first environment contains five ellipsoidal obstacles scattered throughout a 3.2 x 2 workspace domain. The second environment of the same size contains

[1] `http://wiki.ros.org/stdr_simulator`

more general obstacles whose shape cannot be characterized easily by level-sets of closed-form polynomials. In both cases, the robot has no a priori knowledge of the environment and follows a nominal controller that drives it towards a goal point. More formally, we consider a robot with dynamics $\dot{x} = u$, where $x \in \mathcal{D} \subset \mathbb{R}^2$ is the position of the robot and $u \in \mathbb{R}^2$ is the control input. The nominal feedback control policy for all $x \in \mathcal{D}$ is given by $k(x) = \delta \cdot \frac{(x - x_{\text{goal}})}{\|x - x_{\text{goal}}\|}$, where $\delta \in \mathbb{R}_{>0}$, and $x_{\text{goal}} \in \mathcal{D}$ is a desired final goal position for the robot. Informally, the robot must follow $k(x)$ as close as possible while avoiding the unknown obstacles in the workspace. The robot must reach a goal region which is defined as $\mathcal{G} = \{x \in \mathcal{D} \mid \|x - x_{\text{goal}}\| \leq 0.1\}$. For the first scenario, depicted in Fig. 8.2, we obtain ground truth data using a grid-based solution, which is a common approach to compute the true signed distance to the obstacles. The signed distance function corresponds to the true barrier function characterizing the obstacles.

8.5.1 Evaluation Metrics

Comparison of the trajectory outcomes for the different implementations involves two evaluation metrics: the correlation coefficient (R) and the Fréchet distance (F). These metrics capture both the evolutionary mismatch between trajectories, as well as the Euclidean distance mismatch. The combination of both these metrics provides a means to evaluate the outcomes of the proposed algorithms.

Correlation Coefficient

Informally, the correlation coefficient between two trajectories captures the change in one trajectory with respect to the other. That is, one can obtain information regarding the flow of one trajectory with respect to the other. Typically, two trajectories are said to be highly correlated if they have a correlation coefficient greater than 0.7 [131]. In our work, we will consider 0.90 to be the threshold for high correlation between two trajectories. We make use of the correlation coefficient to develop an intuition regarding the nature of the

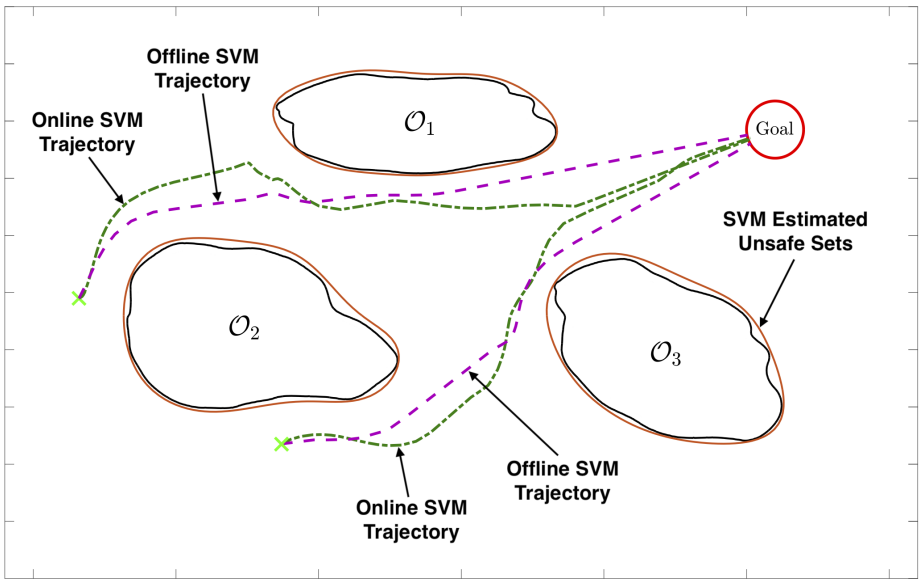


Figure 8.3: An implementation in the STDR simulator where the robot has to navigate the unknown environment to reach a goal region (red circle). Offline kernel-SVM based controller and online kernel-SVM based controller trajectories for two different initial conditions (green crosses) are shown. The obstacles $\mathcal{O}_1$, $\mathcal{O}_2$ and $\mathcal{O}_3$ are such that they cannot be easily characterized by closed form polynomials, and hence, using the traditional CBF formulation is difficult. However, using Algorithm 9 and Algorithm 10, we can generate trajectories such that the robot remains safe.

trajectories generated by the offline and online kernel-SVM based approaches compared with the ground truth data.

Fréchet Distance

Informally, the Fréchet distance provides a measure of the Euclidean distance mismatch between two trajectories. While the correlation coefficient provides information regarding the flow of two trajectories, the Fréchet distance provides an explicit degree of mismatch between the two. A lower Fréchet distance indicates less mismatch between the two trajectories. In particular, $F = 0$ implies that the two trajectories are identical. Lower values of F between two trajectories implies less Euclidean distance mismatch.

8.5.2 Simulation Results

We first consider the five obstacle scenario shown in Fig. 8.2. Two different initial conditions for the robot are considered. Three different trajectories are plotted in each figure. The green dashed trajectory indicates the ground truth trajectory obtained when the barrier function for each obstacle is known a priori. A QP of the form (1.6) is solved to generate this trajectory. The blue, dotted trajectory is generated from the offline kernel-SVM based barrier estimation approach as discussed in Algorithm 9. The purple, dash-dotted trajectory is generated using Algorithm 10 which is the online kernel-SVM based barrier function estimation method. Observe that in both the cases, the robots avoid the obstacle and follow the nominal control policy as close as possible. In the second scenario, we consider a situation where the obstacle shapes are such that finding the closed form expressions for the barrier functions is not straightforward. This setting is as shown in Fig. 8.3. The pink, dashed trajectories are generated using the offline kernel-SVM based barrier function approach as discussed in Algorithm 9, whereas the green, dash-dotted trajectories are generated using the online kernel-SVM based barrier function method described in Algorithm 10. A video of the simulations results is also provided[2].

8.5.3 Discussion & Analysis

Table 8.1 compares the correlation coefficient for both the online and offline approaches against the ground truth trajectory in the first scenario. In addition, the online method is also compared to the offline case. On average, we obtain correlation coefficient values > 0.90, which shows a high similarity between the ground truth trajectory and the barrier estimated trajectory. In particular, note that the average correlation between the offline kernel-SVM approach and the ground truth trajectory is greater then 0.98. We then provide Fréchet distances which measures the degree of mismatch in terms of the Euclidean distance between two 2D trajectories. The smaller the Fréchet distance, the smaller the

[2] `https://youtu.be/-XiaR7QchtQ`

mismatch between the two trajectories. Table 8.2 shows the Fréchet distances between trajectories for the five obstacle scenario. Observe that on average, we obtain distances < 0.10 for each case, which shows that the Euclidean distance mismatch between the trajectories is small. A key inference from the above data is that R_{offline} is very high and F_{offline} is very small, which shows that the offline kernel-SVM estimated barrier function closely replicates the true barrier functions.

8.6 Concluding Remarks

This chapter presented a supervised machine learning based approach to automated synthesis of control barrier functions. A kernel-SVM based method classifies the set of safe and unsafe samples, and generates the desired barrier (level-set) function. A formal guarantee on zero misclassification of unsafe samples is provided along with guarantees on safety of the robot. The proposed framework was evaluated based on the comparison between the generated trajectories and ground truth data. Experimental simulations using the proposed framework were conducted on an omnidirectional robot in a ROS-based simulator using synthetic LiDAR data.

Table 8.1: Correlation Coefficients for Five Obstacle Scenario (Values close to 1 indicate high correlation)

Case	**Offline SVM vs Ground Truth**	**Online SVM vs Ground Truth**	**Offline SVM vs Online SVM**
1	0.9992	0.9777	0.9734
2	0.9627	0.8085	0.8946
3	0.9997	0.9709	0.9694
4	0.9889	0.9466	0.9195
5	0.9954	0.9442	0.9447
6	0.9991	0.9882	0.9870
7	0.9800	0.9865	0.9811
8	0.9692	0.7601	0.5886
9	0.9880	0.8718	0.8665
10	0.9997	0.9899	0.9874
Average	**0.9881**	**0.9244**	**0.9112**

Table 8.2: Fréchet Distance for Five Obstacle Scenario (Smaller values indicate less mismatch between trajectories)

Case	**Offline SVM vs Ground Truth**	**Online SVM vs Ground Truth**	**Offline SVM vs Online SVM**
1	0.0469	0.0822	0.0853
2	0.0665	0.0840	0.1334
3	0.0276	0.0446	0.0468
4	0.0582	0.1444	0.1232
5	0.0660	0.1563	0.1341
6	0.0308	0.0392	0.0266
7	0.1197	0.1296	0.0479
8	0.0496	0.0759	0.0652
9	0.0578	0.1368	0.1119
10	0.0389	0.0327	0.0369
Average	**0.0562**	**0.0925**	**0.0811**

CHAPTER 9
CONCLUSION

A wide range of robotic tasks can be cast as a sequence of operations that must be executed successfully and safely for the task to be satisfied. A simple, yet powerful, example of such a task is lane keeping, where the ego vehicle needs to switch lanes while not colliding with the vehicles ahead or the incoming vehicles in the adjacent lane. Performing such safety critical functions is difficult since we need robust control systems and formal guarantees on safety and task satisfaction. The results in this thesis were aimed to address such a requirement.

In Chapter 3, we proposed a framework for task execution for a robotic system using a combination of zeroing barrier functions and finite time barrier functions. The task was formalized using a tool from formal methods called "Linear Temporal Logic (LTL)". LTL allows for one to capture complex system specifications succinctly. By decomposing the LTL specification into a sequence of reachability problems consisting of target and safety sets, we converted the high level specification into a series of sub-problems that needed to be solved. Then, these sets were encoded using zeroing barrier functions and finite time barrier functions and solved successively. We provided formal guarantees on satisfaction of the given specification as well.

In Chapter 4, we discussed a complex multi-robot scenario where control barrier functions were used in order to smoothly transition between different graph formations corresponding to different behaviors. We provided a centralized and decentralized approach to the problem solution. Then, we discussed key assumptions prevalent in barrier function-based methods which can be problematic when using the frameworks in Chapter 3 and Chapter 4 for safety-critical applications. Hence, the subsequent chapters in the thesis provided techniques to address such assumptions.

The first assumption in the frameworks presented earlier pertain to the feasibility of the optimization program that encodes the barrier functions. Thus in Chapter 5, we illustrated scenarios where the quadratic programs are infeasible, and provided relaxation methods to increase likelihood of feasibility. Along similar lines, in Chapter 6, we presented a higher dimensional barrier function in order to satisfy both position and angular orientation reachability constraints for a differential drive robot. We showed how such tasks can lead to infeasibility of the quadratic program and proposed a new class of barrier functions to resolve the same.

The second assumption relates to the fact that traditional zeroing barrier functions guarantee safety for the system state, which is a point in the state space. However, in reality, since systems have a physical volume associated with them, safety for the volume also needs to be enforced. Hence, in chapter 7, we introduced "Extent-Compatible Control Barrier Functions" which guarantee safety for the system state as well as the volume. In particular, computationally efficient techniques capable of implementation on actual robotic systems were discussed.

The final assumption addresses the fact that in prevalent barrier function-based methods, the system has complete knowledge of the unsafe sets in the domain which may not necessarily be true. We provide a real-life, path planning example to illustrate such a scenario. In Chapter 8, we propose a machine learning and data-driven approach to synthesize control barrier functions. Sensory input from the environment is utilized in order to synthesize a decision boundary i.e. the barrier function. In order to show the workings of the framework, we detailed a path planning task consisting of an omni-directional robot.

There are a number of potential future research directions one could take with regard to the results presented in this thesis. First, the translation of the presented theory to more complex system dynamics and hardware is quintessential for large scale deployment. In particular, systems such as autonomous vehicles will involve a lot more sensors which would need to be integrated efficiently with the frameworks presented in this thesis. Since

such applications involve a lot of data, there are avenues for research in combining embedded processing technology with the techniques presented in this thesis. Second, one could investigate more efficient ways for storing data for the machine learning based approach to barrier function synthesis. Since SVMs are computationally expensive when dealing with large amounts of data and we use a hard margin formulation, the framework could benefit from a learning algorithm that leads to faster barrier function synthesis. This would be particularly useful for the online barrier function synthesis algorithm presented in Chapter 8.

To conclude, there are a number of interesting research directions that can be investigated in order to deploy the techniques presented in this thesis on large scale systems such as autonomous vehicles. With the growing dependency of such systems on sensors and data, an efficient combination of data driven techniques with the frameworks presented in this thesis will result in controllers which guarantee task satisfaction and safety.

REFERENCES

[1] V. James, *Boston dynamics prepares to launch its first commercial robot: Spot*, 2019.

[2] M. Simon, *Your first look inside amazon's robot warehouse of tomorrow*, 2019.

[3] S. Chinchali, S. C. Livingston, U. Topcu, J. W. Burdick, and R. M. Murray, "Towards formal synthesis of reactive controllers for dexterous robotic manipulation," in *2012 IEEE International Conference on Robotics and Automation*, 2012, pp. 5183–5189.

[4] L. P. Kaelbling and T. Lozano-Pérez, "Hierarchical task and motion planning in the now," in *2011 IEEE International Conference on Robotics and Automation*, 2011, pp. 1470–1477.

[5] L. Wang, A. D. Ames, and M. Egerstedt, "Safe certificate-based maneuvers for teams of quadrotors using differential flatness," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, 2017, pp. 3293–3298.

[6] *Trustworthy cyber infrastructure for the power grid (tcip-g)*, 2019.

[7] H. K. Khalil, *Nonlinear systems*, vol. 3.

[8] X. Xu, P. Tabuada, J. W. Grizzle, and A. D. Ames, "Robustness of control barrier functions for safety critical control," *arXiv preprint arXiv:1612.01554*, 2016.

[9] A. Li, L. Wang, P. Pierpaoli, and M. Egerstedt, "Formally correct composition of coordinated behaviors using control barrier certificates," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2018, pp. 3723–3729.

[10] C. Baier and J.-P. Katoen, *Principles of model checking*. 2008.

[11] Meng Guo, K. H. Johansson, and D. V. Dimarogonas, "Revising motion planning under linear temporal logic specifications in partially known workspaces," in *2013 IEEE International Conference on Robotics and Automation*, 2013, pp. 5025–5032.

[12] M. Nagumo, "Über die lage der integralkurven gewöhnlicher differentialgleichungen," *Proceedings of the Physico-Mathematical Society of Japan. 3rd Series*, vol. 24, pp. 551–559, 1942.

[13] S. Prajna, "Barrier certificates for nonlinear model validation," *Automatica*, vol. 42, no. 1, pp. 117–126, Jan. 2006.

[14] S. Prajna and A. Jadbabaie, "Safety verification of hybrid systems using barrier certificates," in *Hybrid Systems: Computation and Control*, R. Alur and G. J. Pappas, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 477–492, ISBN: 978-3-540-24743-2.

[15] S. Prajna and A. Rantzer, "On the necessity of barrier certificates," *IFAC Proceedings Volumes*, vol. 38, no. 1, pp. 526 –531, 2005, 16th IFAC World Congress.

[16] S. Prajna, A. Jadbabaie, and G. J. Pappas, "A framework for worst-case and stochastic safety verification using barrier certificates," *IEEE Transactions on Automatic Control*, vol. 52, no. 8, pp. 1415–1428, 2007.

[17] P. Wieland and F. Allgöwer, "Constructive safety using control barrier functions," *IFAC Proceedings Volumes*, vol. 40, no. 12, pp. 462 –467, 2007, 7th IFAC Symposium on Nonlinear Control Systems.

[18] M. Z. Romdlony and B. Jayawardhana, "Stabilization with guaranteed safety using control lyapunov–barrier function," *Automatica*, vol. 66, pp. 39–47, 2016.

[19] A. D. Ames, J. W. Grizzle, and P. Tabuada, "Control barrier function based quadratic programs with application to adaptive cruise control," in *53rd IEEE Conference on Decision and Control*, 2014, pp. 6271–6278.

[20] A. D. Ames, X. Xu, J. W. Grizzle, and P. Tabuada, "Control barrier function based quadratic programs for safety critical systems," *IEEE Transactions on Automatic Control*, vol. 62, no. 8, pp. 3861–3876, 2017.

[21] A. Mehra, W. Ma, F. Berg, P. Tabuada, J. W. Grizzle, and A. D. Ames, "Adaptive cruise control: Experimental validation of advanced controllers on scale-model cars," in *2015 American Control Conference (ACC)*, 2015, pp. 1411–1418.

[22] W. Xiao, C. Belta, and C. G. Cassandras, "Decentralized merging control in traffic networks: A control barrier function approach," in *Proceedings of the 10th ACM/IEEE International Conference on Cyber-Physical Systems*, ser. ICCPS '19, Montreal, Quebec, Canada: ACM, 2019, pp. 270–279, ISBN: 978-1-4503-6285-6.

[23] L. Wang, A. D. Ames, and M. Egerstedt, "Safe certificate-based maneuvers for teams of quadrotors using differential flatness," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, 2017, pp. 3293–3298.

[24] L. Wang, A. D. Ames, and M. Egerstedt, "Safety barrier certificates for collisions-free multirobot systems," *IEEE Transactions on Robotics*, vol. 33, no. 3, pp. 661–674, 2017.

[25] L. Wang, A. Ames, and M. Egerstedt, "Safety barrier certificates for heterogeneous multi-robot systems," in *2016 American Control Conference (ACC)*, 2016, pp. 5213–5218.

[26] M. Ohnishi, L. Wang, G. Notomista, and M. Egerstedt, "Barrier-certified adaptive reinforcement learning with applications to brushbot navigation," *IEEE Transactions on Robotics*, pp. 1–20, 2019.

[27] L. Wang, D. Han, and M. Egerstedt, "Permissive barrier certificates for safe stabilization using sum-of-squares," in *2018 Annual American Control Conference (ACC)*, 2018, pp. 585–590.

[28] P. Glotfelter, J. Cortés, and M. Egerstedt, "Nonsmooth barrier functions with applications to multi-robot systems," *IEEE Control Systems Letters*, vol. 1, no. 2, pp. 310–315, 2017.

[29] M. Srinivasan, M. Abate, G. Nilsson, and S. Coogan, "Extent-Compatible Control Barrier Functions," *arXiv e-prints*, arXiv:2001.07210, arXiv:2001.07210, Jan. 2020. arXiv: 2001.07210 [eess.SY].

[30] P. Pierpaoli, Li, M. Srinivasan, X. Cai, S. Coogan, and M. Egerstedt, "A Sequential Composition Framework for Coordinating Multi-Robot Behaviors," *arXiv e-prints*, arXiv:1907.07718, arXiv:1907.07718, 2019. arXiv: 1907.07718 [cs.RO].

[31] M. Srinivasan and S. Coogan, "Control of mobile robots using barrier functions under temporal logic specifications," *IEEE Transactions on Robotics*, pp. 1–12, 2020.

[32] M. Srinivasan, S. Coogan, and M. Egerstedt, "Control of multi-agent systems with finite time control barrier certificates and temporal logic," in *2018 IEEE Conference on Decision and Control (CDC)*, 2018, pp. 1991–1996.

[33] L. Guerrero-Bonilla and V. Kumar, "Realization of r-robust formations in the plane using control barrier functions," *IEEE Control Systems Letters*, vol. PP, pp. 1–1, Jun. 2019.

[34] G. Notomista, S. F. Ruf, and M. Egerstedt, "Persistification of robotic tasks using control barrier functions," *IEEE Robotics and Automation Letters*, vol. 3, no. 2, pp. 758–763, 2018.

[35] G. Notomista, S. Mayya, S. Hutchinson, and M. Egerstedt, "An optimal task allocation strategy for heterogeneous multi-robot systems," *CoRR*, vol. abs/1903.08641, 2019. arXiv: 1903.08641.

[36] R. Cheng, G. Orosz, R. M. Murray, and J. W. Burdick, "End-to-end safe reinforcement learning through barrier functions for safety-critical continuous control tasks," *CoRR*, vol. abs/1903.08792, 2019. arXiv: 1903.08792.

[37] S.-C. Hsu, X. Xu, and A. D. Ames, "Control barrier function based quadratic programs with application to bipedal robotic walking," in *2015 American Control Conference (ACC)*, IEEE, 2015, pp. 4542–4548.

[38] W. Xiao and C. Belta, "Control barrier functions for systems with high relative degree," in *2019 IEEE 58th Conference on Decision and Control (CDC)*, 2019, pp. 474–479.

[39] R. Alur, T. A. Henzinger, G. Lafferriere, and G. J. Pappas, "Discrete abstractions of hybrid systems," *Proceedings of the IEEE*, vol. 88, no. 7, pp. 971–984, 2000.

[40] C. Belta and L. C. G. J. M. Habets, "Controlling a class of nonlinear systems on rectangles," *IEEE Transactions on Automatic Control*, vol. 51, no. 11, pp. 1749–1759, 2006.

[41] M. Kloetzer and C. Belta, "A fully automated framework for control of linear systems from temporal logic specifications," *IEEE Transactions on Automatic Control*, vol. 53, no. 1, pp. 287–297, 2008.

[42] T. Wongpiromsarn, U. Topcu, and R. M. Murray, "Receding horizon temporal logic planning," *IEEE Transactions on Automatic Control*, vol. 57, no. 11, pp. 2817–2830, 2012.

[43] A. Bhatia, M. R. Maly, L. E. Kavraki, and M. Y. Vardi, "Motion planning with complex goals," *IEEE Robotics Automation Magazine*, vol. 18, no. 3, pp. 55–64, 2011.

[44] G. E. Fainekos, A. Girard, H. Kress-Gazit, and G. J. Pappas, "Temporal logic motion planning for dynamic robots," *Automatica*, vol. 45, no. 2, pp. 343–352, 2009.

[45] E. M. Wolff and R. M. Murray, "Optimal control of nonlinear systems with temporal logic specifications," in *ISRR*, 2013.

[46] E. M. Wolff, U. Topcu, and R. M. Murray, "Optimization-based control of nonlinear systems with linear temporal logic specifications."

[47] E. M. Wolff, "Control of dynamical systems with temporal logic specifications," PhD thesis, California Institute of Technology, 2014.

[48] L. Lindemann and D. V. Dimarogonas, "Control barrier functions for signal temporal logic tasks," *IEEE control systems letters*, vol. 3, no. 1, pp. 96–101, 2019.

[49] V. Raman, A. Donzé, M. Maasoumy, R. M. Murray, A. Sangiovanni-Vincentelli, and S. A. Seshia, "Model predictive control with signal temporal logic specifications," in *53rd IEEE Conference on Decision and Control*, 2014, pp. 81–87.

[50] C. Belta and S. Sadraddini, "Formal methods for control synthesis: An optimization perspective," *Annual Review of Control, Robotics, and Autonomous Systems*, 2019.

[51] Z. Liu, B. Wu, J. Dai, and H. Lin, "Distributed communication-aware motion planning for multi-agent systems from stl and spatel specifications," in *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*, 2017, pp. 4452–4457.

[52] S. S. Farahani, R. Majumdar, V. S. Prabhu, and S. E. Z. Soudjani, "Shrinking horizon model predictive control with chance-constrained signal temporal logic specifications," in *2017 American Control Conference (ACC)*, 2017, pp. 1740–1746.

[53] D. Aksaray, A. Jones, Z. Kong, M. Schwager, and C. Belta, "Q-learning for robust satisfaction of signal temporal logic specifications," in *2016 IEEE 55th Conference on Decision and Control (CDC)*, 2016, pp. 6565–6570.

[54] D. Muniraj, K. G. Vamvoudakis, and M. Farhood, "Enforcing signal temporal logic specifications in multi-agent adversarial environments: A deep q-learning approach," in *2018 IEEE Conference on Decision and Control (CDC)*, IEEE, 2018, pp. 4141–4146.

[55] P. Várnai and D. V. Dimarogonas, "Prescribed performance control guided policy improvement for satisfying signal temporal logic tasks," *CoRR*, vol. abs/1903.04340, 2019. arXiv: 1903.04340.

[56] L. Lindemann and D. V. Dimarogonas, "Decentralized robust control of coupled multi-agent systems under local signal temporal logic tasks," in *2018 Annual American Control Conference (ACC)*, IEEE, 2018, pp. 1567–1573.

[57] Y. V. Pant, H. Abbas, R. A. Quaye, and R. Mangharam, "Fly-by-logic: Control of multi-drone fleets with temporal logic objectives," in *2018 ACM/IEEE 9th International Conference on Cyber-Physical Systems (ICCPS)*, 2018, pp. 186–197.

[58] A. Robey, H. Hu, L. Lindemann, H. Zhang, D. V. Dimarogonas, S. Tu, and N. Matni, "Learning Control Barrier Functions from Expert Demonstrations," *arXiv*

e-prints, arXiv:2004.03315, arXiv:2004.03315, Apr. 2020. arXiv: 2004.03315 [eess.SY].

[59] W. Jin, Z. Wang, Z. Yang, and S. Mou, "Neural Certificates for Safe Control Policies," *arXiv e-prints*, arXiv:2006.08465, arXiv:2006.08465, Jun. 2020. arXiv: 2006.08465 [eess.SY].

[60] R. Cheng, G. Orosz, R. M. Murray, and J. W. Burdick, "End-to-end safe reinforcement learning through barrier functions for safety-critical continuous control tasks," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, 2019, pp. 3387–3395.

[61] A. Taylor, A. Singletary, Y. Yue, and A. Ames, "Learning for safety-critical control with control barrier functions," in *Learning for Dynamics and Control*, PMLR, 2020, pp. 708–717.

[62] M. Maghenem and R. G. Sanfelice, "Barrier function certificates for forward invariance in hybrid inclusions," in *2018 IEEE Conference on Decision and Control (CDC)*, 2018, pp. 759–764.

[63] L. Lindemann and D. V. Dimarogonas, "Decentralized control barrier functions for coupled multi-agent systems under signal temporal logic tasks," in *2019 18th European Control Conference (ECC)*, 2019, pp. 89–94.

[64] E. M. Wolff, U. Topcu, and R. M. Murray, "Efficient reactive controller synthesis for a fragment of linear temporal logic," in *2013 IEEE International Conference on Robotics and Automation*, 2013, pp. 5033–5040.

[65] H. Kress-Gazit, G. E. Fainekos, and G. J. Pappas, "Temporal-logic-based reactive mission and motion planning," *IEEE Transactions on Robotics*, vol. 25, no. 6, pp. 1370–1381, 2009.

[66] T. Wongpiromsarn, U. Topcu, and A. Lamperski, "Automata theory meets barrier certificates: Temporal logic verification of nonlinear systems," *IEEE Transactions on Automatic Control*, vol. 61, no. 11, pp. 3344–3355, 2016.

[67] D. Pickem, P. Glotfelter, L. Wang, M. Mote, A. Ames, E. Feron, and M. Egerstedt, "The robotarium: A remotely accessible swarm robotics research testbed," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2017, pp. 1699–1706.

[68] P. Glotfelter, I. Buckley, and M. Egerstedt, "Hybrid nonsmooth barrier functions with applications to provably safe and composable collision avoidance for robotic systems," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 1303–1310, 2019.

[69] R. Olfati-Saber, "Near-identity diffeomorphisms and exponential ϵ-tracking and ϵ-stabilization of first-order nonholonomic SE(2) vehicles," in *Proceedings of the 2002 American Control Conference (IEEE Cat. No.CH37301)*, vol. 6, 2002, 4690–4695 vol.6.

[70] S. Kim, K. Jang, S. Park, Y. Lee, S. Y. Lee, and J. Park, "Continuous task transition approach for robot controller based on hierarchical quadratic programming," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 1603–1610, 2019.

[71] G. Jarquín, A. Escande, G. Arechavaleta, T. Moulard, E. Yoshida, and V. Parra-Vega, "Real-time smooth task transitions for hierarchical inverse kinematics," in *2013 13th IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, 2013, pp. 528–533.

[72] S. Boyd and L. Vandenberghe, *Convex optimization*. Cambridge university press, 2004.

[73] R. Konda, A. D. Ames, and S. Coogan, *Characterizing Safety: Minimal Barrier Functions from Scalar Comparison Systems*, Under review. `https://arxiv.org/abs/1908.09323`, 2019.

[74] M. Srinivasan, N. P. Hyun, and S. Coogan, "Weighted polar finite time control barrier functions with applications to multi-robot systems," in *2019 IEEE 58th Conference on Decision and Control (CDC)*, 2019, pp. 7031–7036.

[75] J. Cortés and M. Egerstedt, "Coordinated control of multi-robot systems: A survey," *SICE Journal of Control, Measurement, and System Integration*, vol. 10, no. 6, pp. 495–503, 2017.

[76] J. Lin, A. S. Morse, and B. D. Anderson, "The multi-agent rendezvous problem," in *42nd IEEE International Conference on Decision and Control (IEEE Cat. No. 03CH37475)*, IEEE, vol. 2, 2003, pp. 1508–1513.

[77] W. Ren, R. W. Beard, and T. W. McLain, "Coordination variables and consensus building in multiple vehicle systems," in *Cooperative control*, Springer, 2005, pp. 171–188.

[78] J. L. Ramirez, M. Pavone, and E. Frazzoli, "Cyclic pursuit for spacecraft formation control," in *Proceedings of the American Control Conference*, 2009, pp. 4811–4817.

[79] J. R. Lawton, R. W. Beard, and B. J. Young, "A decentralized approach to formation maneuvers," *IEEE transactions on robotics and automation*, vol. 19, no. 6, pp. 933–941, 2003.

[80] I. Buckley and M. Egerstedt, "Infinitesimally shape-similar motions using relative angle measurements," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2017, pp. 1077–1082.

[81] J. Cortes, S. Martinez, T. Karatas, and F. Bullo, "Coverage control for mobile sensing networks," *IEEE Transactions on robotics and Automation*, vol. 20, no. 2, pp. 243–255, 2004.

[82] M. Santos, Y. Diaz-Mercado, and M. Egerstedt, "Coverage control for multirobot teams with heterogeneous sensing capabilities," *IEEE Robotics and Automation Letters*, vol. 3, no. 2, pp. 919–925, 2018.

[83] M. Mesbahi and M. Egerstedt, *Graph theoretic methods in multiagent networks*. Princeton University Press, 2010, vol. 33.

[84] H. G. Tanner, A. Jadbabaie, and G. J. Pappas, "Flocking in fixed and switching networks," *IEEE Transactions on Automatic control*, vol. 52, no. 5, pp. 863–868, 2007.

[85] S. Nagavalli, N. Chakraborty, and K. Sycara, "Automated sequencing of swarm behaviors for supervisory control of robotic swarms," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2017, pp. 2674–2681.

[86] R. K. Ramachandran, J. A. Preiss, and G. S. Sukhatme, "Resilience by reconfiguration: Exploiting heterogeneity in robot teams," *arXiv preprint arXiv:1903.04856*, 2019.

[87] P. Culbertson and M. Schwager, "Decentralized adaptive control for collaborative manipulation," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2018, pp. 278–285.

[88] A. D. Ames, S. Coogan, M. Egerstedt, G. Notomista, K. Sreenath, and P. Tabuada, "Control barrier functions: Theory and applications," *2019 European Control Conference (ECC)*, pp. 3420–3431, 2019.

[89] L. Wang, A. D. Ames, and M. Egerstedt, "Multi-objective compositions for collision-free connectivity maintenance in teams of mobile robots," in *2016 IEEE 55th Conference on Decision and Control (CDC)*, IEEE, 2016, pp. 2659–2664.

[90] E. Squires, P. Pierpaoli, R. Konda, S. Coogan, and M. Egerstedt, "Composition of safety constraints with applications to decentralized fixed-wing collision avoidance," *arXiv preprint arXiv:1906.03771*, 2019.

[91] R. K. Williams and G. S. Sukhatme, "Observability in topology-constrained multi-robot target tracking," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2015, pp. 1795–1801.

[92] J. Aspnes, T. Eren, D. K. Goldenberg, A. S. Morse, W. Whiteley, Y. R. Yang, B. D. Anderson, and P. N. Belhumeur, "A theory of network localization," *IEEE Transactions on Mobile Computing*, vol. 5, no. 12, pp. 1663–1678, 2006.

[93] J. Wagenpfeil, A. Trachte, T. Hatanaka, M. Fujita, and O. Sawodny, "Distributed decision making for task switching via a consensus-like algorithm," in *2009 American Control Conference*, IEEE, 2009, pp. 5761–5766.

[94] P. Twu, P. Martin, and M. Egerstedt, "Graph process specifications for hybrid networked systems," *IFAC Proceedings Volumes*, vol. 43, no. 12, pp. 65–70, 2010.

[95] "Military operations in urbanized terrain," *US Army Field Manual 90-10*, 1975.

[96] A. D. Ames, X. Xu, J. W. Grizzle, and P. Tabuada, "Control barrier function based quadratic programs for safety critical systems," *IEEE Transactions on Automatic Control*, vol. 62, no. 8, pp. 3861–3876, 2017.

[97] G. Notomista, S. Mayya, S. Hutchinson, and M. Egerstedt, "An optimal task allocation strategy for heterogeneous multi-robot systems," *arXiv preprint arXiv:1903.08641*, 2019.

[98] L. Wang, A. D. Ames, and M. Egerstedt, "Safety barrier certificates for collisions-free multirobot systems," *IEEE Transactions on Robotics*, vol. 33, no. 3, pp. 661–674, 2017.

[99] L. Wang, A. Ames, and M. Egerstedt, "Safety barrier certificates for heterogeneous multi-robot systems," in *2016 American Control Conference (ACC)*, 2016, pp. 5213–5218.

[100] J. Tůmová, L. I. Reyes Castro, S. Karaman, E. Frazzoli, and D. Rus, "Minimum-violation ltl planning with conflicting specifications," in *2013 American Control Conference*, 2013, pp. 200–205.

[101] J. Fink, M. A. Hsieh, and V. Kumar, "Multi-robot manipulation via caging in environments with obstacles," in *2008 IEEE International Conference on Robotics and Automation*, 2008, pp. 1471–1476.

[102] A. W. Stroupe, M. C. Martin, and T. Balch, "Distributed sensor fusion for object position estimation by multi-robot systems," in *Proceedings 2001 ICRA. IEEE International Conference on Robotics and Automation (Cat. No.01CH37164)*, vol. 2, 2001, 1092–1098 vol.2.

[103] D. Shishika and V. Kumar, "Local-game decomposition for multiplayer perimeter-defense problem," in *2018 IEEE Conference on Decision and Control (CDC)*, 2018, pp. 2093–2100.

[104] N.-s. P. Hyun, P. A. Vela, and E. I. Verriest, "Bendable Cuboid Robot Path Planning with Collision Avoidance using Generalized L_p Norms," arXiv:1712.06021, arXiv:1712.06021, 2017. arXiv: `1712.06021 [math.OC]`.

[105] N. P. Hyun, P. A. Vela, and E. I. Verriest, "A new framework for optimal path planning of rectangular robots using a weighted l_p norm," *IEEE Robotics and Automation Letters*, vol. 2, no. 3, pp. 1460–1465, 2017.

[106] A. H. Chang, N. P. Hyun, E. I. Verriest, and P. A. Vela, "Optimal trajectory planning and feedback control of lateral undulation in snake-like robots," in *2018 Annual American Control Conference (ACC)*, 2018, pp. 2114–2120.

[107] A. Papachristodoulou, J. Anderson, S. P. G. Valmorbida, P. Seiler, and P. A. Parrilo, *SOSTOOLS: Sum of squares optimization toolbox for MATLAB*, `http://arxiv.org/abs/1310.4716`, 2013.

[108] K. K. Hauser, "Minimum constraint displacement motion planning.," in *Robotics: Science and Systems*, 2013.

[109] L. Wang, A. D. Ames, and M. Egerstedt, "Safe certificate-based maneuvers for teams of quadrotors using differential flatness," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, 2017, pp. 3293–3298.

[110] O. Arslan and D. E. Koditschek, "Sensor-based reactive navigation in unknown convex sphere worlds," *The International Journal of Robotics Research*, vol. 38, no. 2-3, pp. 196–223, 2019.

[111] M. Srinivasan, A. Dabholkar, S. Coogan, and P. Vela, "Synthesis of control barrier functions using a supervised machine learning approach," *arXivpreprint: 2003.04950*, 2020.

[112] P. Vlantis, C. Vrohidis, C. P. Bechlioulis, and K. J. Kyriakopoulos, "Orientation-aware motion planning in complex workspaces using adaptive harmonic potential fields," in *2019 International Conference on Robotics and Automation (ICRA)*, 2019, pp. 8592–8598.

[113] A. D. Ames, S. Coogan, M. Egerstedt, G. Notomista, K. Sreenath, and P. Tabuada, "Control barrier functions: Theory and applications," in *2019 18th European Control Conference (ECC)*, 2019, pp. 3420–3431.

[114] L. Vandenberghe and S. Boyd, "Semidefinite programming," *SIAM review*, vol. 38, no. 1, pp. 49–95, 1996.

[115] R. Hussain and S. Zeadally, "Autonomous cars: Research results, issues, and future challenges," *IEEE Communications Surveys Tutorials*, vol. 21, no. 2, pp. 1275–1313, 2019.

[116] G. Beltrame, E. Merlo, J. Panerati, and C. Pinciroli, "Engineering safety in swarm robotics," in *2018 IEEE/ACM 1st International Workshop on Robotics Software Engineering (RoSE)*, 2018, pp. 36–39.

[117] J. C. Knight, "Safety critical systems: Challenges and directions," in *Proceedings of the 24th International Conference on Software Engineering*, ser. ICSE '02, Orlando, Florida: Association for Computing Machinery, 2002, 547–550, ISBN: 158113472X.

[118] L. Wang, A. D. Ames, and M. Egerstedt, "Safety barrier certificates for collisions-free multirobot systems," *IEEE Transactions on Robotics*, vol. 33, no. 3, pp. 661–674, 2017.

[119] L. Wang, D. Han, and M. Egerstedt, "Permissive barrier certificates for safe stabilization using sum-of-squares," in *2018 Annual American Control Conference (ACC)*, 2018, pp. 585–590.

[120] S. M. Richards, F. Berkenkamp, and A. Krause, "The lyapunov neural network: Adaptive stability certification for safe learning of dynamical systems," in *Proceedings of the 2nd Conference on Robot Learning (CoRL)*, PMLR, vol. 87, 2018, pp. 466–476.

[121] R. Cheng, G. Orosz, R. M. Murray, and J. W. Burdick, "End-to-end safe reinforcement learning through barrier functions for safety-critical continuous control tasks," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, 2019, pp. 3387–3395.

[122] N. Cristianini, J. Shawe-Taylor, *et al.*, *An introduction to support vector machines and other kernel-based learning methods*. Cambridge university press, 2000.

[123] F. Ramos and L. Ott, "Hilbert maps: Scalable continuous occupancy mapping with stochastic gradient descent," *The International Journal of Robotics Research*, vol. 35, no. 14, pp. 1717–1730, 2016.

[124] G. Francis, L. Ott, and F. Ramos, "Fast stochastic functional path planning in occupancy maps," in *International Conference on Robotics and Automation*, 2019, pp. 929–935.

[125] T. Duong, N. Das, M. Yip, and N. Atanasov, "Autonomous Navigation in Unknown Environments using Sparse Kernel-based Occupancy Mapping," *arXiv e-prints*, arXiv:2002.01921, arXiv:2002.01921, 2020. arXiv: 2002.01921 [cs.RO].

[126] *Lidar comparison chart, https://autonomoustuff.com/lidar-chart/.*

[127] T. Gurriet, M. Mote, A. D. Ames, and E. Feron, "An online approach to active set invariance," in *2018 IEEE Conference on Decision and Control (CDC)*, 2018, pp. 3592–3599.

[128] K. Veropoulos, C. Campbell, N. Cristianini, *et al.*, "Controlling the sensitivity of support vector machines," in *Proceedings of the international joint conference on AI*, vol. 55, 1999, p. 60.

[129] B. Hammer and K. Gersmann, "A note on the universal approximation capability of support vector machines," *neural processing letters*, vol. 17, no. 1, pp. 43–53, 2003.

[130] S. Ross, G. Gordon, and D. Bagnell, "A reduction of imitation learning and structured prediction to no-regret online learning," in *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, 2011, pp. 627–635.

[131] Y. J. Kim, S. W. Park, H. G. Yeom, M. S. Bang, J. S. Kim, C. K. Chung, and S. Kim, "A study on a robot arm driven by three-dimensional trajectories predicted from non-invasive neural signals," *Biomedical engineering online*, vol. 14, no. 1, p. 81, 2015.